**Assignment - 6**

1) (Provide only mathematical solutions for this question) Six points with the following attributes are given, calculate and find out clustering representations and dendrogram using Single, complete, and average link proximity function in hierarchical clustering technique.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import scipy.cluster.hierarchy as shc

from scipy.spatial.distance import squareform, pdist

a = np.array([0.4005,0.2148,0.3457,0.2652,0.0789,0.4548])

b = np.array([0.5306,0.3854,0.3156,0.1875,0.4139,0.3022])

point = ['P1','P2','P3','P4','P5','P6']

data = pd.DataFrame({'Point':point, 'x cordinate':a, 'y cordinate':b})

data = data.set\_index('Point')

data

dist = pd.DataFrame(squareform(np.round(pdist(data[['x cordinate', 'y cordinate']]),4), 'euclidean'), columns=data.index.values, index=data.index.values)

dist

plt.figure(figsize=(10,4))

plt.title("Dendrogram with Single inkage")

dend = shc.dendrogram(shc.linkage(data[['x cordinate', 'y cordinate']], method='single'), labels=data.index)

plt.figure(figsize=(10,4))

plt.title("Dendrogram with Complete inkage")

dend = shc.dendrogram(shc.linkage(data[['x cordinate', 'y cordinate']], method='complete'), labels=data.index)

plt.figure(figsize=(10,4))

plt.title("Dendrogram with Average inkage")

dend = shc.dendrogram(shc.linkage(data[['x cordinate', 'y cordinate']], method='average'), labels=data.index)

2) Use CC\_GENERAL.csv given in the folder and apply: a) Preprocess the data by removing the categorical column and filling the missing values. b) Apply StandardScaler() and normalize() functions to scale and normalize raw input data. c) Use PCA with K=2 to reduce the input dimensions to two features. d) Apply Agglomerative Clustering with k=2,3,4 and 5 on reduced features and visualize result for each k value using scatter plot. e) Evaluate different variations using Silhouette Scores and Visualize results with a bar chart.

#importing all libraries here for assignment

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import preprocessing,metrics

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.decomposition import PCA

from sklearn.cluster import AgglomerativeClustering

from sklearn.metrics import silhouette\_score

import warnings

warnings.filterwarnings("ignore")

dataframe = pd.read\_csv('CC GENERAL.csv')

dataframe.info()

dataframe.head()

dataframe.describe()

df = dataframe.drop(['CUST\_ID'], axis=1)

df.head()

df.isnull().any()

df.fillna(dataframe.mean(), inplace=True)

df.isnull().any()

df.corr().style.background\_gradient(cmap="Greens")

x = df.iloc[:,0:-1]

y = df.iloc[:,-1]

scaler = preprocessing.StandardScaler()

scaler.fit(x)

X\_scaled\_array = scaler.transform(x)

X\_scaled\_df = pd.DataFrame(X\_scaled\_array, columns = x.columns)

#Normalization is the process of scaling individual samples to have unit norm.

#This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

X\_normalized = preprocessing.normalize(X\_scaled\_df)

# Converting the numpy array into a pandas DataFrame

X\_normalized = pd.DataFrame(X\_normalized)

pca2 = PCA(n\_components=2)

principalComponents = pca2.fit\_transform(X\_normalized)

principalDf = pd.DataFrame(data = principalComponents, columns = ['P1', 'P2'])

finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)

finalDf.head()

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

plt.scatter(finalDf['P1'],finalDf['P2'],c=finalDf['TENURE'],cmap='prism', s =5)

plt.xlabel('pc1')

plt.ylabel('pc2')

ac2 = AgglomerativeClustering(n\_clusters = 2)

# Visualizing the clustering

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

plt.scatter(principalDf['P1'], principalDf['P2'],

c = ac2.fit\_predict(principalDf), cmap ='rainbow')

plt.show()

ac3 = AgglomerativeClustering(n\_clusters = 3)

# Visualizing the clustering

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

plt.scatter(principalDf['P1'], principalDf['P2'],

c = ac3.fit\_predict(principalDf), cmap ='rainbow')

plt.show()

ac4 = AgglomerativeClustering(n\_clusters = 4)

# Visualizing the clustering

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

plt.scatter(principalDf['P1'], principalDf['P2'],

c = ac4.fit\_predict(principalDf), cmap ='rainbow')

plt.show()

ac5 = AgglomerativeClustering(n\_clusters = 5)

# Visualizing the clustering

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

plt.scatter(principalDf['P1'], principalDf['P2'],

c = ac5.fit\_predict(principalDf), cmap ='rainbow')

plt.show()

k = [2, 3, 4, 5]

# Appending the silhouette scores of the different models to the list

silhouette\_scores = []

silhouette\_scores.append(

silhouette\_score(principalDf, ac2.fit\_predict(principalDf)))

silhouette\_scores.append(

silhouette\_score(principalDf, ac3.fit\_predict(principalDf)))

silhouette\_scores.append(

silhouette\_score(principalDf, ac4.fit\_predict(principalDf)))

silhouette\_scores.append(

silhouette\_score(principalDf, ac5.fit\_predict(principalDf)))

# Plotting a bar graph to compare the results

plt.bar(k, silhouette\_scores)

plt.xlabel('Number of clusters', fontsize = 20)

plt.ylabel('S(i)', fontsize = 20)

plt.show()

**https://github.com/NXI57230/Assignment1\_700725723**