INTELLIGENT SYSTEMS

LAB-3

NAME: UMANG TIWARI

SEC: A

REG NO.: 201800591

ROLL: 53

PROBLEM STATEMENT

Write a program to explore the use of unsupervised learning methods for clustering data, and also for obtaining lower dimensional representations.

1. Submit the pdf file with code and graphs (py, doc file will not be evaluated)

2. Analyze the algorithm by varying number of clusters (k) and features at least for 2 types of dataset

3. Determine the number of clusters using the Elbow method and plot the graph.

PROBLEM SOLUTION

SOURCE CODE

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.datasets.\_samples\_generator import make\_blobs

from sklearn.cluster import KMeans

from scipy.spatial.distance import cdist

sns.set()

*# First Dataset*

X, Y = make\_blobs(n\_samples = 300, centers = 3, cluster\_std = 0.60, random\_state = 0)

plt.scatter(X[:, 0], X[:, 1], c = Y, s = 10, cmap = "Accent")

plt.show()

kmeans = KMeans(n\_clusters = 3)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

centers = kmeans.cluster\_centers\_

plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=10, cmap='viridis')

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=100, alpha=0.9);

plt.show()

distortions = []

inertias = []

mapping1 = {}

mapping2 = {}

for k in range(1, 10):

    kmeanModel = KMeans(n\_clusters = k).fit(X)

    kmeanModel.fit(X)

    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster\_centers\_, 'euclidean'), axis = 1)) / X.shape[0])

    inertias.append(kmeanModel.inertia\_)

    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster\_centers\_, 'euclidean'), axis = 1)) / X.shape[0]

    mapping2[k] = kmeanModel.inertia\_

for key, val in mapping1.items():

    print(f'{key} : {val}')

plt.plot(range(1, 10), distortions, 'bx-')

plt.xlabel('Values of K')

plt.ylabel('Distortion')

plt.title('The Elbow Method using Distortion')

plt.show()

*# Second Dataset*

x = np.array([3, 1, 1, 2, 1, 6, 6, 6, 5, 6, 7, 8, 9, 8, 9, 9, 8])

y = np.array([5, 4, 5, 6, 5, 8, 6, 7, 6, 7, 1, 2, 1, 2, 3, 2, 3])

X = np.array(list(zip(x, y))).reshape(len(x), 2)

plt.scatter(x, y, s = 25)

plt.show()

kmeans = KMeans(n\_clusters = 3)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

centers = kmeans.cluster\_centers\_

plt.scatter(X[:, 0], X[:, 1], c = y\_kmeans, s = 25, cmap = 'viridis')

plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 100, alpha = 0.9);

plt.show()

distortions = []

inertias = []

mapping1 = {}

mapping2 = {}

for k in range(1, 10):

    kmeanModel = KMeans(n\_clusters = k).fit(X)

    kmeanModel.fit(X)

    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster\_centers\_, 'euclidean'), axis = 1)) / X.shape[0])

    inertias.append(kmeanModel.inertia\_)

    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster\_centers\_, 'euclidean'), axis = 1)) / X.shape[0]

    mapping2[k] = kmeanModel.inertia\_

for key, val in mapping1.items():

    print(f'{key} : {val}')

plt.plot(range(1, 10), distortions, 'bx-')

plt.xlabel('Values of K')

plt.ylabel('Distortion')

plt.title('The Elbow Method using Distortion')

plt.show()

OUTPUT:

Performing necessary imports:

Graphical user interface, text, application

Description automatically generated

Loading the dataset

Chart, scatter chart

Description automatically generated

Applying K-Means clusteringGraphical user interface, text, application

Description automatically generated

Chart, scatter chart

Description automatically generated

For k value = 4

Chart, scatter chart

Description automatically generated

Number of clusters using elbow method



Second dataset

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

For k = 2

Chart, scatter chart

Description automatically generated