# **M9 Lab: Clustering Methods in SciKit Learn**

import pandas as pd;

df = pd.read\_csv("./Country-data.csv")

df

Graphical user interface

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**import sklearn libraries**

import numpy as np

from sklearn.cluster import KMeans

X = df.iloc[:,1:].copy()#.apply(lambda x: (x-np.min(x))/(np.max(x)-np.min(x)))

KMC1 = KMeans(n\_clusters=2, random\_state=0).fit(X)

res1 = df.iloc[:,:1].copy()

res1.insert(1,'Cluster\_ID',KMC1.labels\_)

res1.sort\_values(by='Cluster\_ID')

res1.sort\_values(by='Cluster\_ID').to\_csv('KMEANS\_2.csv')

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

1. Open this csv file & see which countries are grouped together into the same cluster.

Do you notice anything interesting about how the different countries are grouped into clusters?

Clustering is done based on the income value.

for K in [3,4,6]:

KMC1 = KMeans(n\_clusters=K, random\_state=0).fit(X)

res1 = df.iloc[:,:1].copy()

res1.insert(1,'Cluster\_ID',KMC1.labels\_)

# res1.sort\_values(by='Cluster\_ID')

res1.sort\_values(by='Cluster\_ID').to\_csv(f'KMEANS\_{K}.csv')

Graphical user interface, text, email

Description automatically generated

## Agglomerative Clustering

from sklearn.cluster import AgglomerativeClustering

AC1 = AgglomerativeClustering(n\_clusters=2).fit(X)

AC1.labels\_

Ares1 = df.iloc[:,:1].copy()

Ares1.insert(1,'Cluster\_ID',AC1.labels\_)

Ares1.sort\_values(by='Cluster\_ID')

Ares1.sort\_values(by='Cluster\_ID').to\_csv('Agglo\_2.csv')

Graphical user interface, text, application

Description automatically generated

#### **Repeat for other cluster size**

for K in [3,4,6]:

print(f'Cluster size:{K}')

AC1 = AgglomerativeClustering(n\_clusters=K).fit(X)

Ares1 = df.iloc[:,:1].copy()

Ares1.insert(1,'Cluster\_ID',AC1.labels\_)

Ares1.sort\_values(by='Cluster\_ID').to\_csv(f'Agglo\_{K}.csv')

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

**DBSCAN**

from sklearn.cluster import DBSCAN

import numpy as np

clustering = DBSCAN(eps=800, min\_samples=3).fit(X)

clustering.labels\_

np.unique(clustering.labels\_,return\_counts=True)

Graphical user interface, text, application, email

Description automatically generated

* What is the number of obtained clusters?

The number of cluster is 5 (0,1,2,3,4)

* How many data points are considered as noise?

84 data points are considered as niose.

for eps in range(700,1000,100):

for min\_samples in range(2,4,1):

clustering = DBSCAN(eps=eps, min\_samples=min\_samples).fit(X)

print(f"eps={eps} & min\_sample={min\_samples}: {np.unique(clustering.labels\_,return\_counts=True)}\n")

Graphical user interface, text, application

Description automatically generated

* How do you assess the obtained clustering results as compared to the output generated by KMEANS & Agglomerative Clustering?

Noise or outliers are ignored in case of DBScan algorithm. In case of Kmeans & Agglomerative Clustering all are considered for clustering.

* Are the obtained results for these values different or similar to each other?

Obtained results are different for different radius values and the min\_samples.

For example : minimum sample =2 and radius = 700.

There are around 9 clusters and 74 noise points.

When compared with same radius and minimum sample = 3

There are around 5 clusters and 84 noise points.

Graphical user interface

Description automatically generated