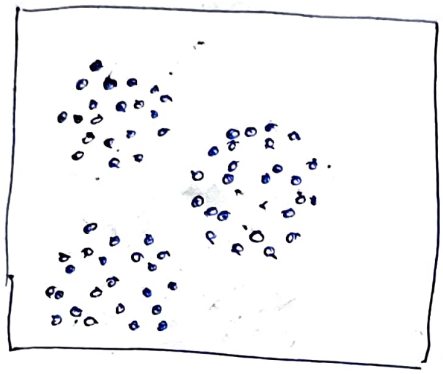
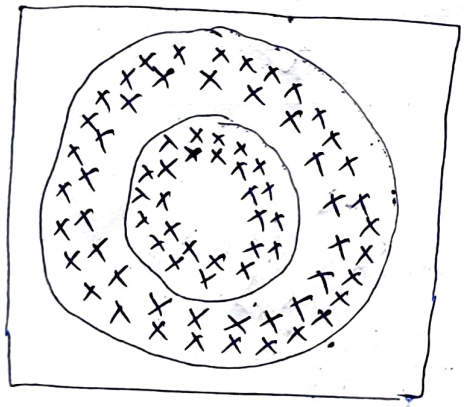
Hierarchical clustering

Drawbacks of K-Means clustering

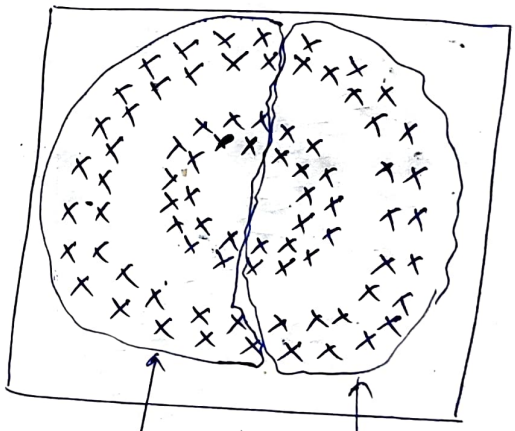
1) K-means relies on distance blw pt to centroid.
So K means works for below data very well



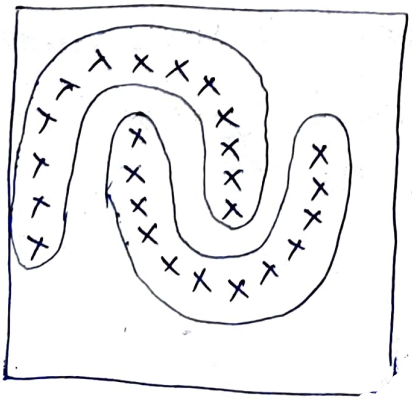
But doesn't work well with below shown datasets



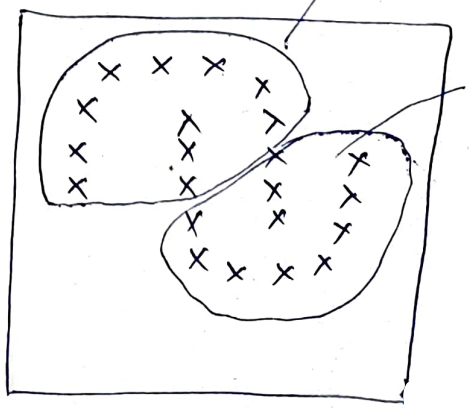
clustering using K-means



This is wrong clustering as it should have made inner cluster/circle as one and outer circle as second cluster.

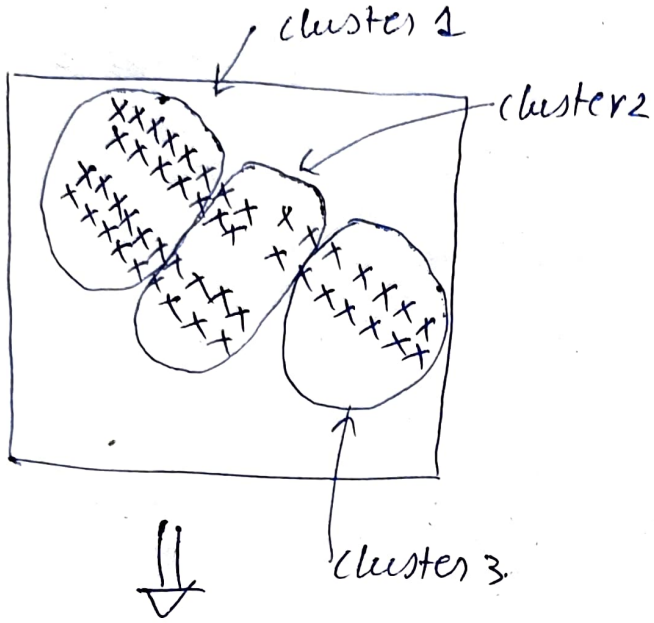


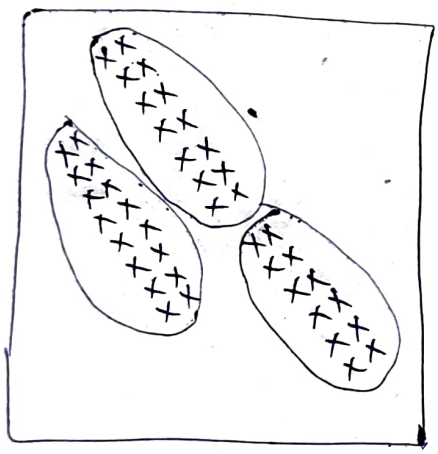
clustering using K-means



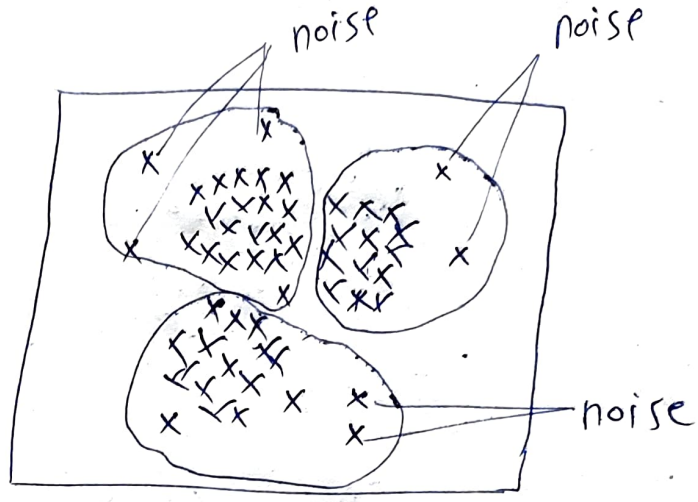
In above example k-mean fails to correctly group the data and clusters generated are completely wrong.

And same thing is done with below given datasets

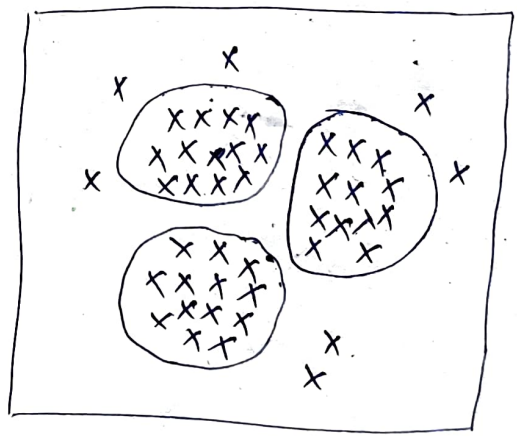




Correct clustering



noise/ dataset
outlier



correct clustering

Here the algorithm excludes noise/outliers.

Here the algorithm is DB scan.

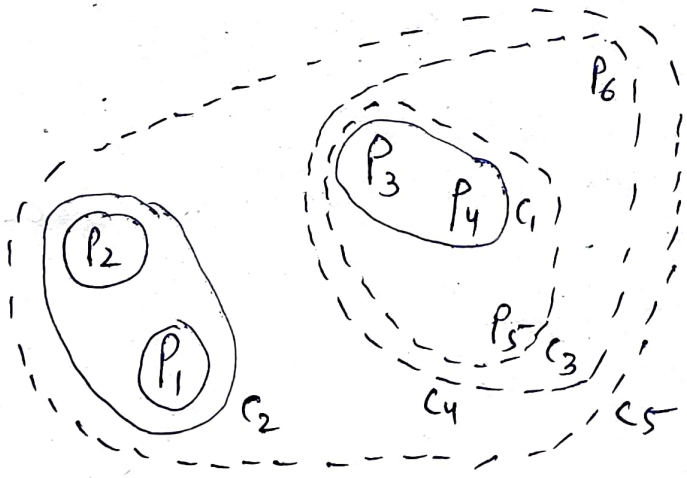
Types of Hierarchical clustering

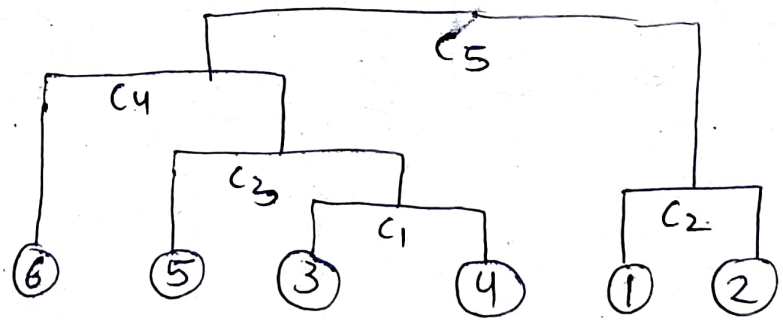
1) Agglomerative clustering
2) Divisive clustering (This is not used on large scale)

Agglomerative Clustering

→ Here we assume each data point as individual cluster in starting.

→ we will calculate dist blw each cluster and cluster having minimum dist will be merged and made a single cluster as shown in C_1 cluster.





and we will make dandogram as shown above for ③ and ④ point and name it as C_1 cluster.

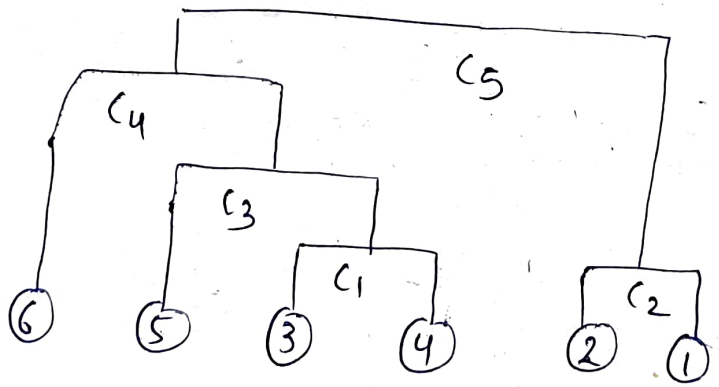
After that we will go for next minimum dist say pt ① and ② we will make them a cluster C_2 and add to dendogram.

Next minimum distance is of point ⑤ with cluster C_1, so we will make a new cluster C_3 and add it to dendogram.

Now point ⑥ is close to cluster C_3 so we will make new cluster C_4 with cluster C_3 and point ⑥ and add to dendogram.

Finally cluster C_2 and C_4 will be merged into single cluster C_5 and added to dendogram.

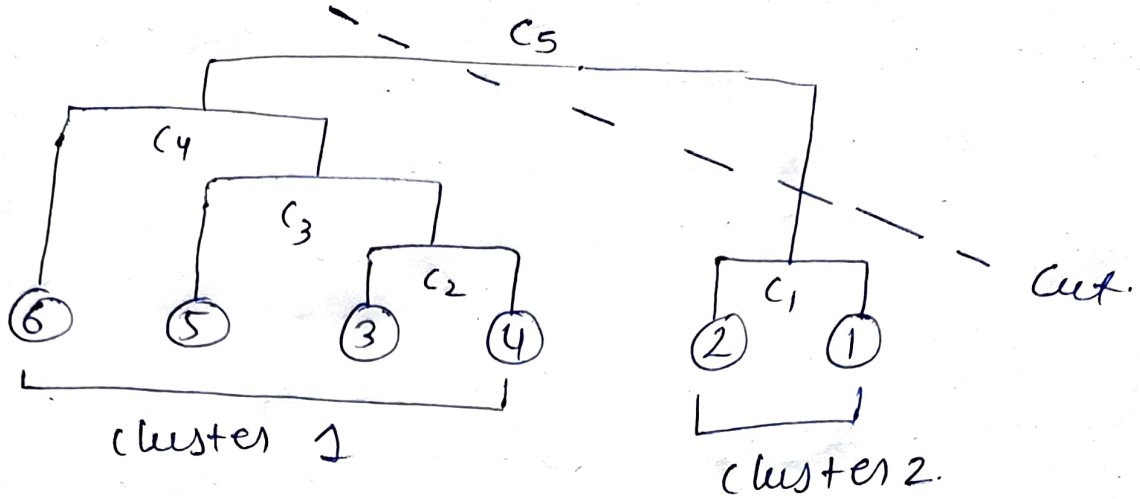
The final dendogram will look as shown below.



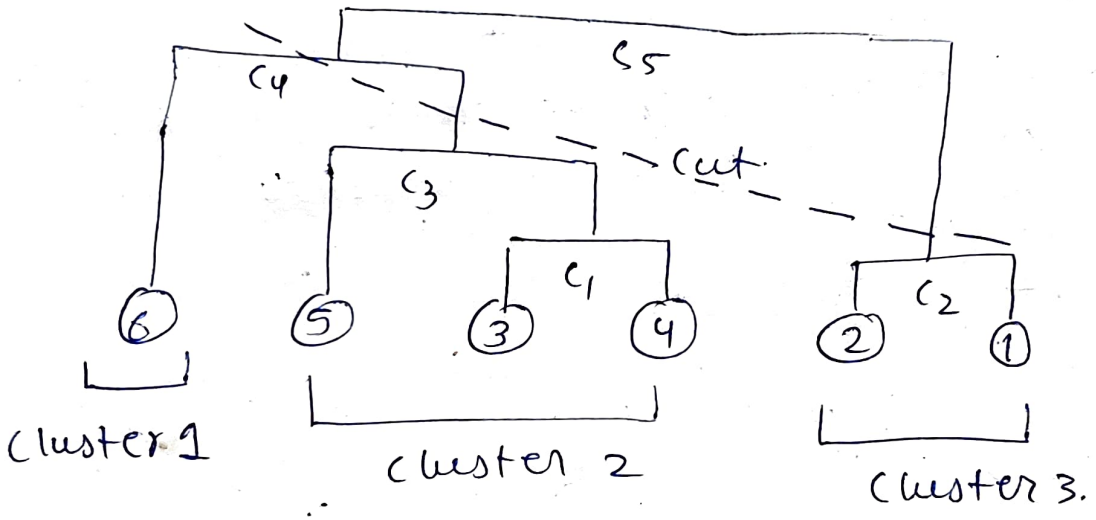
⇐ Dendogram

Now initially we had 6 clusters ie 6 points.

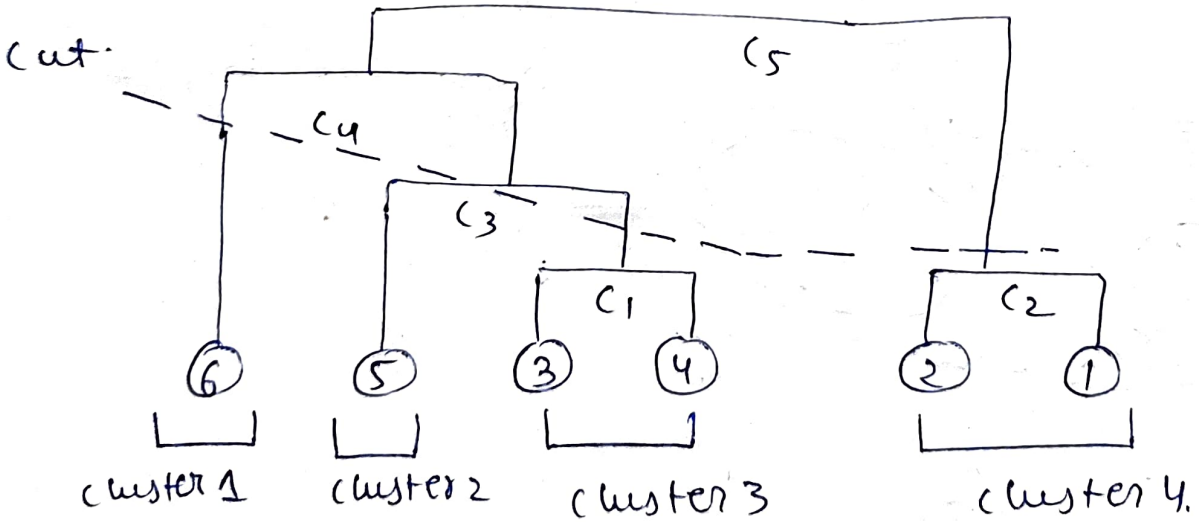
So say now we want 2 clusters.



3 clusters



4 clusters



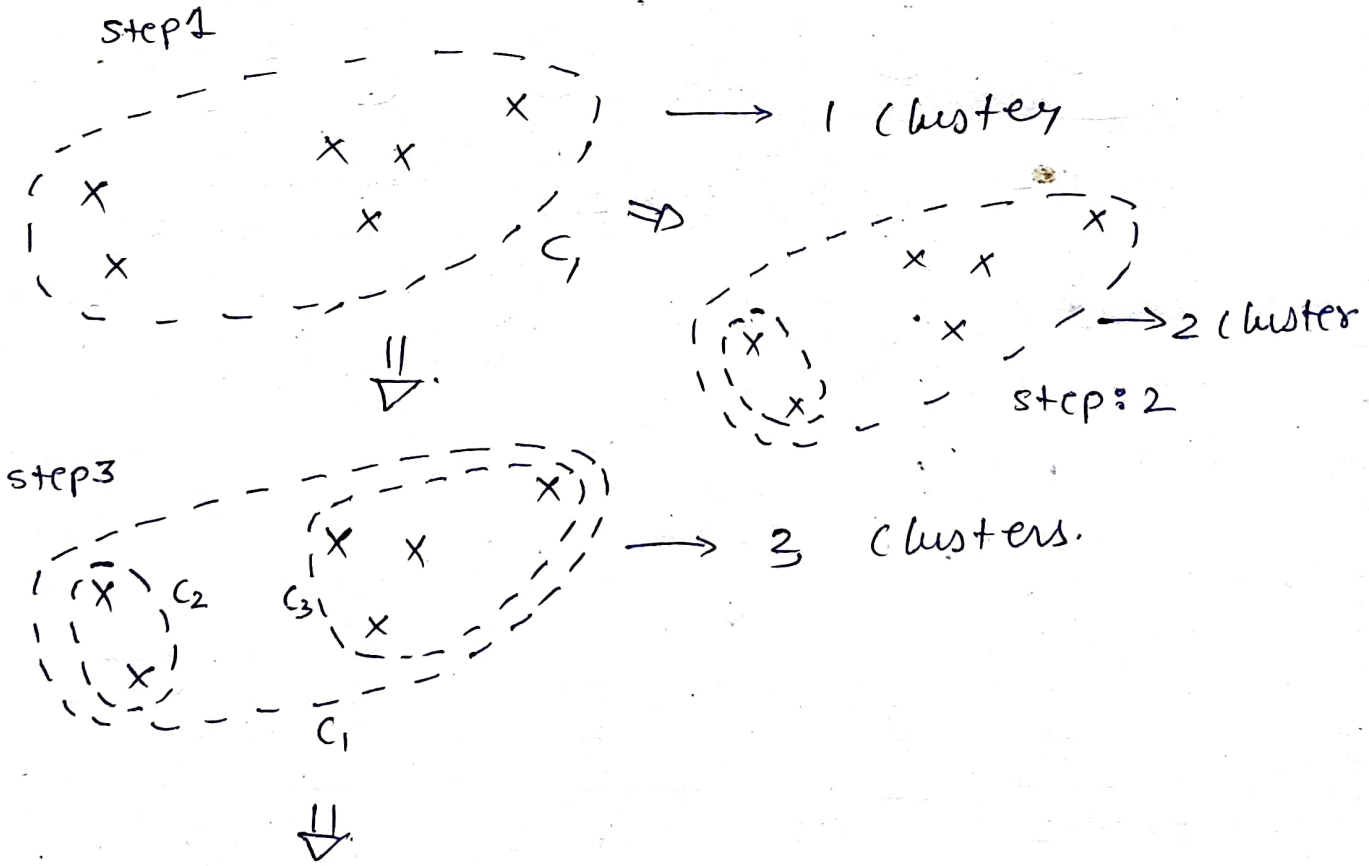
Divisive clustering

→ In this approach it will consider the whole dataset as single cluster.

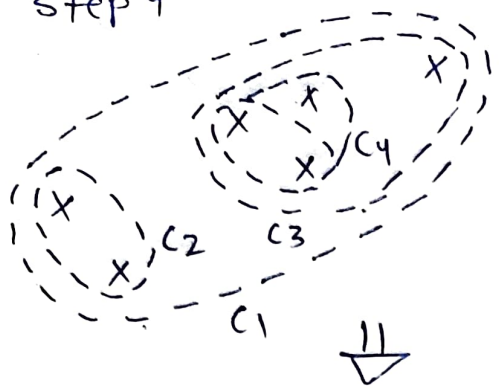
→ In next step it will try to divide whole data set into two clusters based on distance (min).

→ It will try to increase No. of cluster in each iteration. This will continue till each datapoint is individual cluster.

→ For example say we start with 6 datapoints as shown below. So at first there will be a single cluster consisting of all 6 datapoints



Step 4



→ 4 clusters

→ So on Till we have individual data points as a cluster.

* How agglomerative clustering Algorithm works?

1) Initialize the proximity matrix.

→ proximity matrix is square matrix which has no. of rows and colums equal to data point count. ie if 5 data point then we will have 5x5 matrix.

→ The proximity matrix has distance stored in it wrt two individual points

2) make each point a Cluster and calculate the distances of point wrt each other and store in proximity matrix.

3) Inside a loop

1) merge two points wrt smallest distance in proximity matrix and form a cluster.

2) create new proximity matrix with new cluster and remaining points. update the distance of clusters with each points.

3) when only two clusters are left then final distance is calculated using 4 different techniques which we will see below.

4) Repeat the whole process till only one cluster is left.

* lets see above with an example

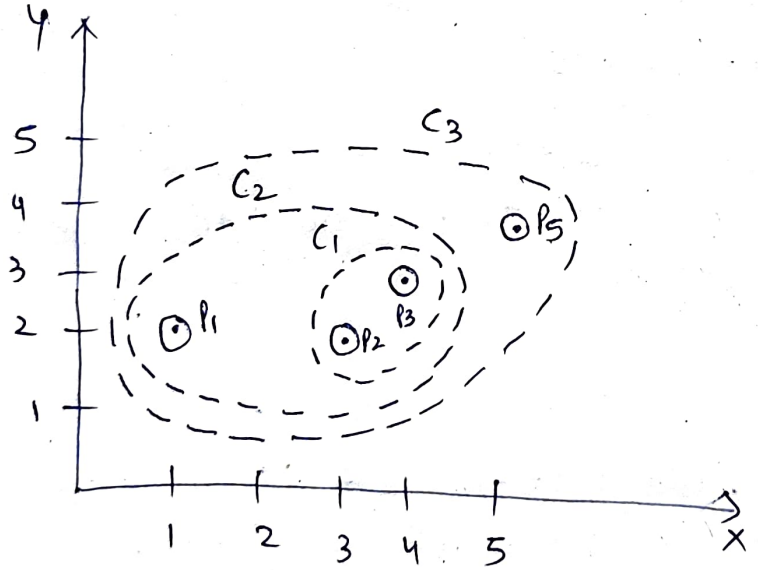
4 Random points.

$P_1(5,4)$, $P_2(3,2)$ $P_3(4,3)$ $P_4(1,2)$

	P_1	P_2	P_3	P_4
P_1	0			
P_2	2.83	0		
P_3	1.41	1.41	0	
P_4	4.47	2	3.16	0

Here we have two options to make first cluster ie P_1 and P_3 & P_2 and P_3

lets select P_2 and P_3 randomly for cluster 1



	P_1	C_1	P_4
P_1	0		
C_1	2.12	0	
P_4	4.47	2.91	0

for distance calculation purpose calculating mean of points in C_1 cluster.

$C_1 \Rightarrow \left(\frac{3+4}{2}, \frac{3+2}{2}\right) \Rightarrow (3.5, 2.5)$

So C_2 will be combination of C_1 and P_1

for C_2 cluster

$$\left(\frac{3.5+5}{2}, \frac{2.5+4}{2}\right) \Rightarrow (4.25, 3.25)$$

	C_2	P_4
C_2	0	
P_4	3.48	0

Now the final cluster will be C_3 which is formed after merging P_4 and C_2.

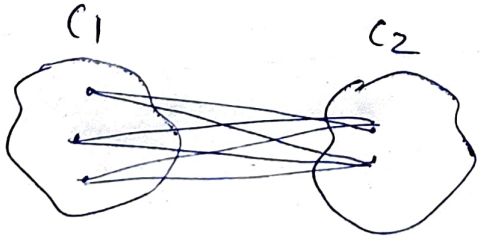
Types of Agglomerative Clustering

1) Min (single Link)
2) Max (complete Link)
3) Average.
4) Ward.

} Inter cluster distance/ similarity

① single link (min)

Here we calculate distance blw all points of two clusters and choose the shortest distance.



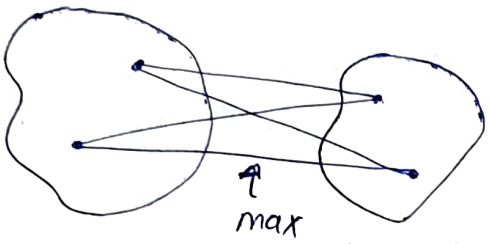
Total $3 \times 2 \Rightarrow 6$ distances

And shortest distance will be selected.

→ This works well with data without outliers.

② Max (complete link)

Here also we calculate distance b/w all points in two clusters and select maximum distance

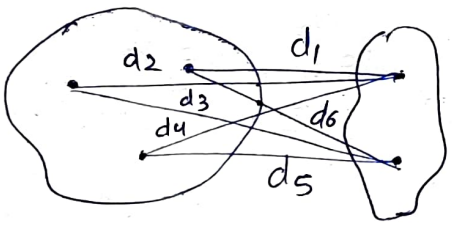


Total distances = 2 x 2 = 4.

→ This works well with outliers/Noise.

→ This doesn't work well with imbalanced dataset

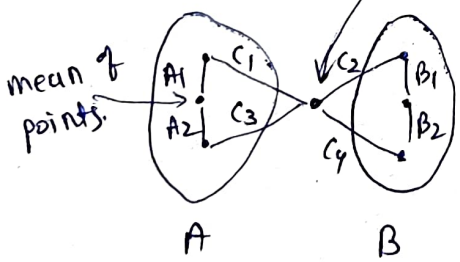
③ Group Average



$$\text{Group Average} = \frac{d_1 + d_2 + \dots + d_6}{6}$$

6 ← 2 x 3.

④ WARD

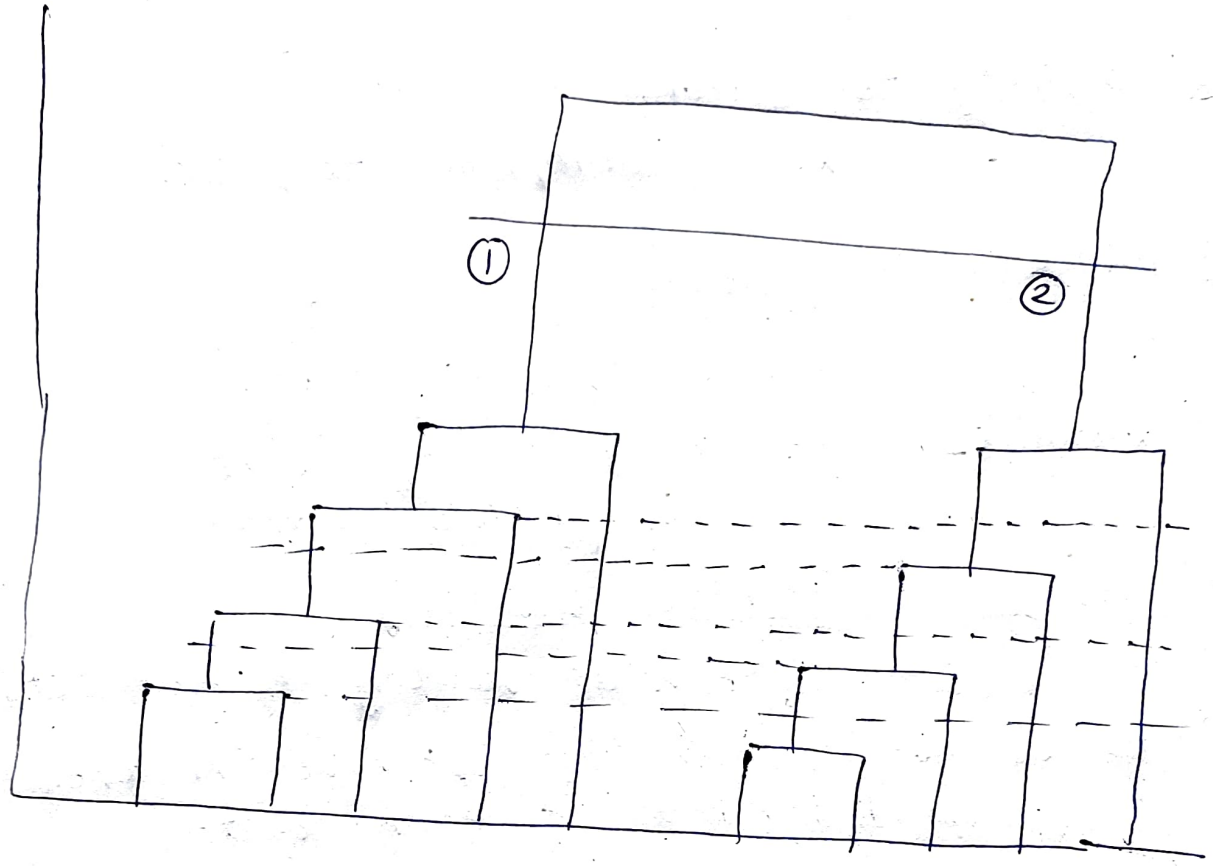


1) First calculate centroid of all points in A and B

$$(dist)_{A-B} = C_1^2 + C_2^2 + C_3^2 + C_4^2 - A_1^2 - A_2^2 - B_1^2 - B_2^2$$

* How to find ideal No. of clusters.

This is done using dendogram.



① Draw dendogram

② Find that vertical line which is not cut by any horizontal line (dotted line as shown)

③ one the vertical line is identified draw a horizontal, solid line. The points at which this horizontal line cuts vertical line, count these point

The total count of these points is equal to No. of clusters

* vertical line is basically inter cluster distance/ similarity

Hyperparameter

① n_clusters => signifies no. of clusters

② affinity => distance calculation methods
Ex: Euclidean, manhattan, etc.

③ linkage => Type of agglomeratic
Eg: ward, max, min, etc.

④ Distance_threshold => minimum inter cluster distance required to merge two clusters

Benefits of Hierarchical clustering

1) can be used with wide variety of datasets
2) As we plot dendogram, we have the knowledge of nearest neighbour of each point.

Limitations

1) cannot be used with large data set due to use of proximity matrix. size for large data set

Suppose we have 10^6 data point

=> proximity matrix size will be $10^6 \times 10^6$ ie 10^{12} bytes of data

which is very large for memory.

Hierarchical Clustering

1.0 Importing required libraries

In [1]:
```
import pandas as pd
import numpy as np

from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import LabelEncoder

import scipy.cluster.hierarchy as hc

import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

2.0 Importing customer dataset for clustering

In [2]:
```
dataset=pd.read_csv('segmented_customers.csv')
dataset.head()
```

Out[2]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	1	1	19	15	39	3
1	2	1	21	15	81	4
2	3	0	20	16	6	3
3	4	0	23	16	77	4
4	5	0	31	17	40	3

In [3]:
```
data=dataset.iloc[:,3:5]
data.head()
```

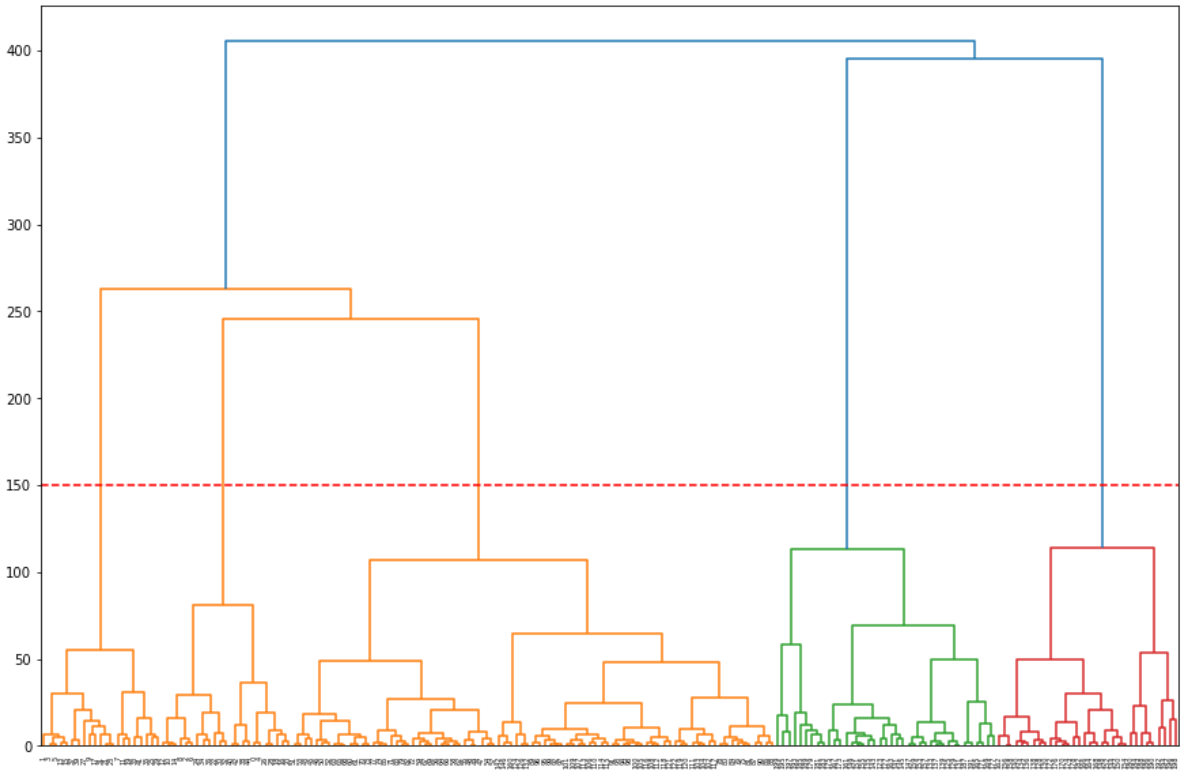
Out[3]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

2.1 Plotting dendogram for deciding number of cluster

From below dendogram it is clearly visible that there will be 5 clusters as when we draw a line parallel to X axis at Y= 150, it cuts 5 vertical lines.

In [5]:
```
plt.figure(figsize=(15,10))
hc.dendrogram(hc.linkage(data, method='ward'))
plt.axhline(y=150, color='r', linestyle='--')
plt.show();
```



2.2 Clustering the data

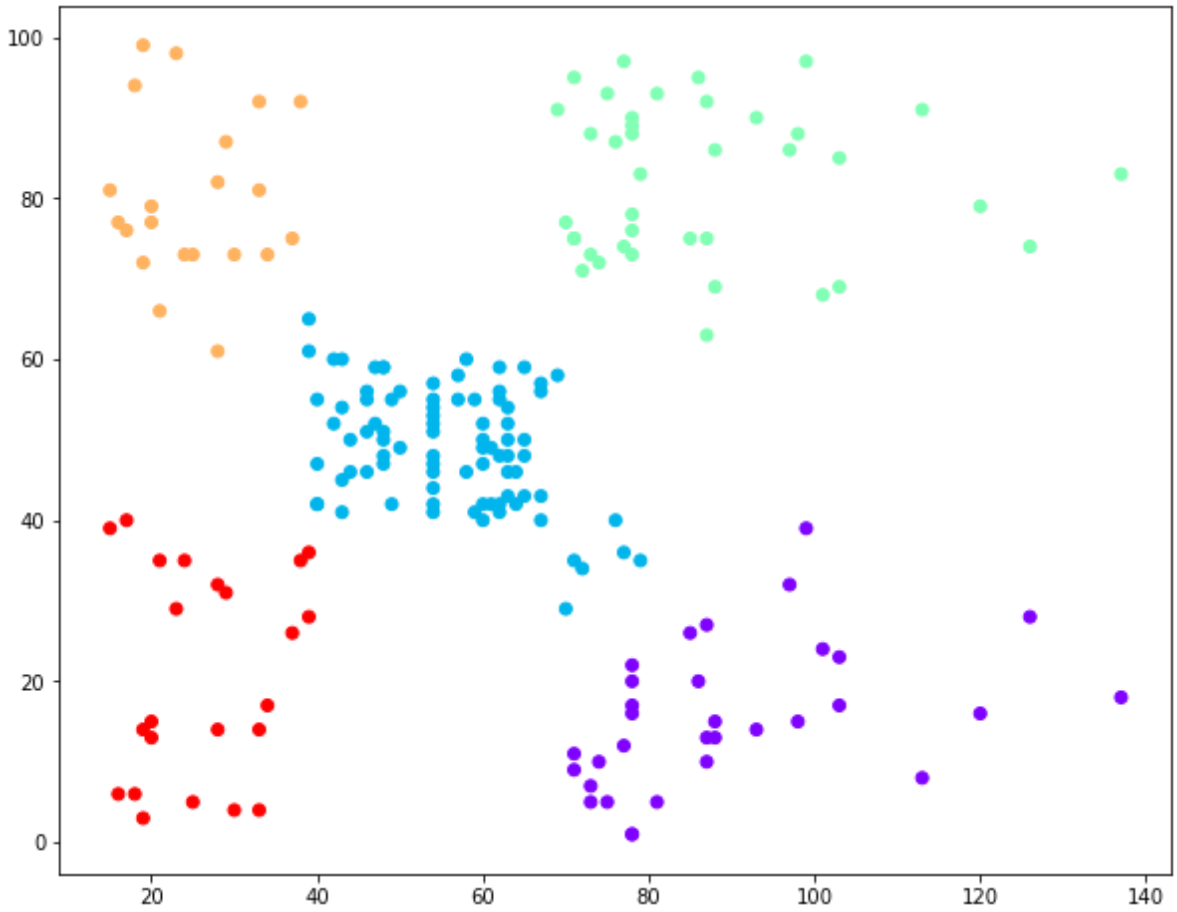
In [6]:
```
agg_cluster= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='
labels_= agg_cluster.fit_predict(data)
labels_
```

Out[6]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 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, 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, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int64)
```

2.3 Visualizing the clustered data

In [7]:
```
plt.figure(figsize=(10,8))
plt.scatter(data['Annual Income (k$)'], data['Spending Score (1-100)'], c=agg_clust
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x23f664189a0>
```



3.0 Importing World Countries dataset for clustering

In [140…
```
data1= pd.read_csv('countries of the world.csv')
data1.head()
```

Out[140]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDI cap
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48,0	0,00	23,06	163,07	7(
1	Albania	EASTERN EUROPE	3581655	28748	124,6	1,26	-4,93	21,52	45(
2	Algeria	NORTHERN AFRICA	32930091	2381740	13,8	0,04	-0,39	31	60(
3	American Samoa	OCEANIA	57794	199	290,4	58,29	-20,71	9,27	80(
4	Andorra	WESTERN EUROPE	71201	468	152,1	0,00	6,6	4,05	190(

3.1 Data Cleaning

In [141…
```
### getting count of unique values in each feature
for feature in data1.columns:
    print(f"The [{feature}] and has [{data1[feature].nunique()}] no. of unique valu
```

```
The [Country] and has [227] no. of unique values
The [Region] and has [11] no. of unique values
The [Population] and has [227] no. of unique values
The [Area (sq. mi.)] and has [226] no. of unique values
The [Pop. Density (per sq. mi.)] and has [219] no. of unique values
The [Coastline (coast/area ratio)] and has [151] no. of unique values
The [Net migration] and has [157] no. of unique values
The [Infant mortality (per 1000 births)] and has [220] no. of unique values
The [GDP ($ per capita)] and has [130] no. of unique values
The [Literacy (%)] and has [140] no. of unique values
The [Phones (per 1000)] and has [214] no. of unique values
The [Arable (%)] and has [203] no. of unique values
The [Crops (%)] and has [162] no. of unique values
The [Other (%)] and has [209] no. of unique values
The [Climate] and has [6] no. of unique values
The [Birthrate] and has [220] no. of unique values
The [Deathrate] and has [201] no. of unique values
The [Agriculture] and has [150] no. of unique values
The [Industry] and has [155] no. of unique values
The [Service] and has [167] no. of unique values
```

In [142…
```
### replacing ',' with '.' in features
for feature in [feature for feature in data1.columns if data1[feature].dtypes=='O'
    try:
        data1[feature]=data1[feature].str.replace(',','.')
    except:
        pass
```

In [143…
```
### getting no of columns and rows in dataset
print(f"Data set has {data1.shape[1]} no. of columns and {data1.shape[0]} no. of r
```

```
Data set has 20 no. of columns and 227 no. of rows
```

In [144…
```
### creating label encoding object
encoder=LabelEncoder()
encoder
```

Out[144]:
```
▾ LabelEncoder

LabelEncoder()
```

In [145…
```
### getting datatypes
data1.dtypes
```

Out[145]:
```
Country                                object
Region                                 object
Population                              int64
Area (sq. mi.)                          int64
Pop. Density (per sq. mi.)             object
Coastline (coast/area ratio)           object
Net migration                          object
Infant mortality (per 1000 births)     object
GDP ($ per capita)                    float64
Literacy (%)                           object
Phones (per 1000)                      object
Arable (%)                             object
Crops (%)                              object
Other (%)                              object
Climate                                object
Birthrate                              object
Deathrate                              object
Agriculture                            object
Industry                               object
Service                                object
dtype: object
```

In [146…
```
### converting numerical data to numeric datatypes
for feature in data1.columns:
    if data1[feature].dtypes in ['int64', 'float64']:
        pass
    else:
        try:
            data1[feature]=data1[feature].astype('float64')
        except:
            pass
```

In [147…
```
### checking datatypes
data1.dtypes
```

Out[147]:
```
Country                                object
Region                                 object
Population                              int64
Area (sq. mi.)                          int64
Pop. Density (per sq. mi.)            float64
Coastline (coast/area ratio)          float64
Net migration                         float64
Infant mortality (per 1000 births)    float64
GDP ($ per capita)                    float64
Literacy (%)                          float64
Phones (per 1000)                     float64
Arable (%)                            float64
Crops (%)                             float64
Other (%)                             float64
Climate                               float64
Birthrate                             float64
Deathrate                             float64
Agriculture                           float64
Industry                              float64
Service                               float64
dtype: object
```

In [149…
```
### label encoding the categorical features
data1 = data1.apply(encoder.fit_transform)
```

In [152…
```
### getting top 5 rows
data1.head()
```

Out[152]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Litera ('
0	0	0	189	185	75	0	156	218	2	
1	1	3	98	84	138	52	17	111	34	
2	2	6	190	215	27	3	69	138	45	
3	3	8	19	13	179	129	1	63	56	1
4	4	10	25	31	153	0	143	9	93	1

In [151…
```
### checking null values in dataset
data1.isnull().sum()
```

Out[151]:
```
Country                               0
Region                                0
Population                            0
Area (sq. mi.)                        0
Pop. Density (per sq. mi.)            0
Coastline (coast/area ratio)          0
Net migration                         0
Infant mortality (per 1000 births)    0
GDP ($ per capita)                    0
Literacy (%)                          0
Phones (per 1000)                     0
Arable (%)                            0
Crops (%)                             0
Other (%)                             0
Climate                               0
Birthrate                             0
Deathrate                             0
Agriculture                           0
Industry                              0
Service                               0
dtype: int64
```

3.2 Clustering Countries on Birthrate

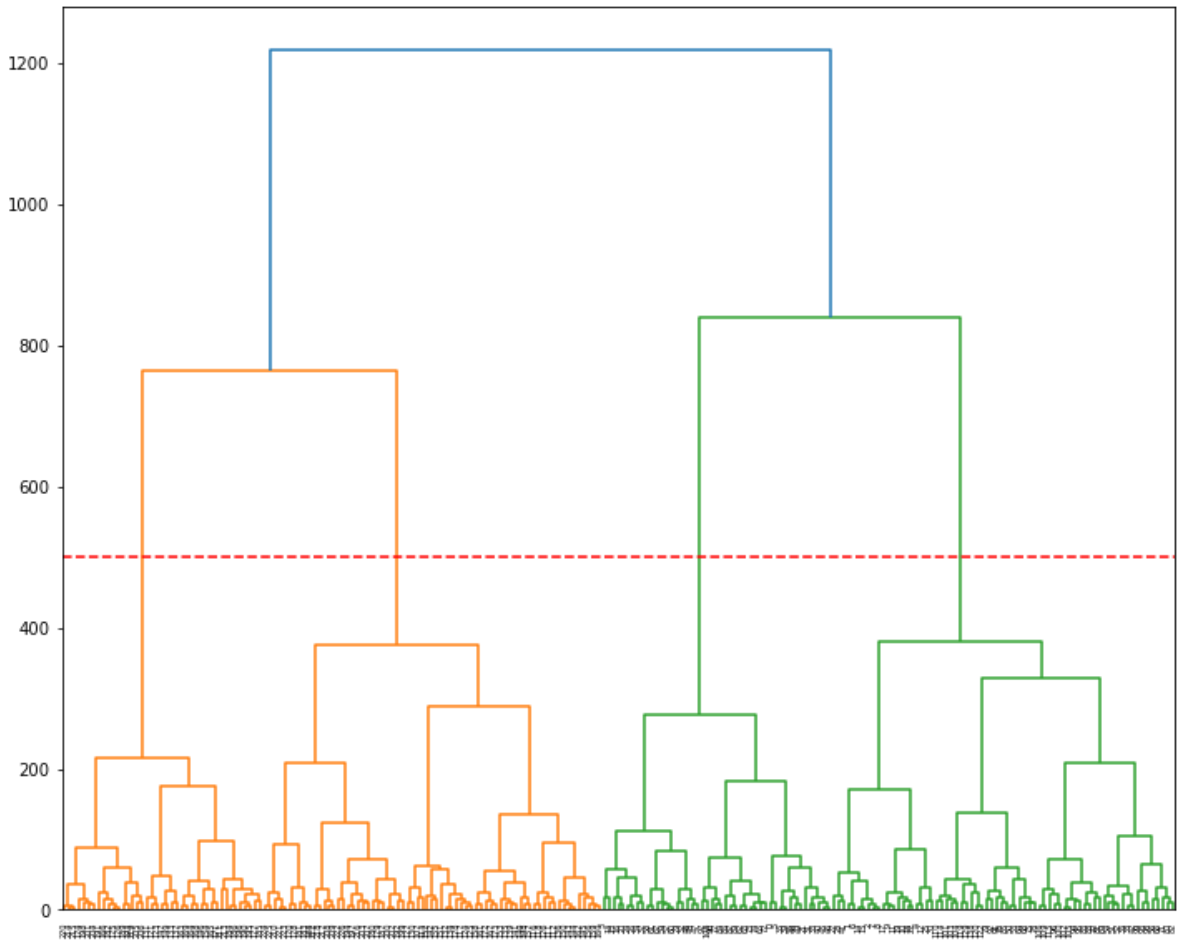
In [99]:
```
### creating a copy of data set with 3 features for clustering
data2=data1[['Country', 'Region', 'Birthrate']]
data2.head()
```

Out[99]:

	Country	Region	Birthrate
0	0	0	216
1	1	3	74
2	2	6	95
3	3	8	131
4	4	10	3

In [101…
```
### plotting a dendogram to get count of clusters and it comes out to be 4
plt.figure(figsize=(12,10))
hc.dendrogram(hc.linkage(data2, method='ward'))
```

```
plt.axhline(y=500, color='r', linestyle='--')
plt.show();
```

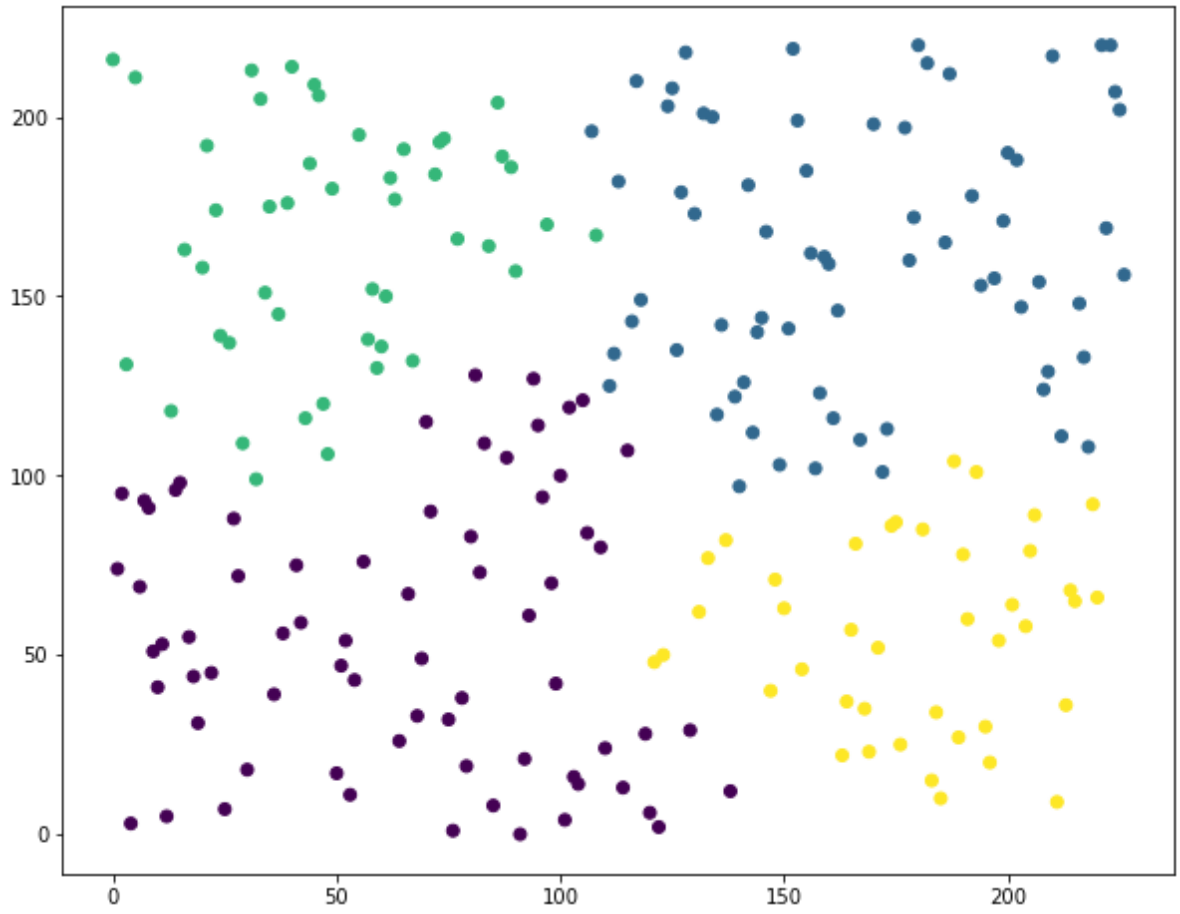


In [102…
```
agg_cluster= AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='
labels_= agg_cluster.fit_predict(data2)
labels_
```

Out[102]:
```
array([2, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 2, 2,
       0, 2, 2, 0, 2, 0, 0, 2, 0, 2, 2, 2, 2, 2, 0, 2, 0, 2, 2, 0, 0, 2,
       2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 2,
       0, 2, 0, 0, 0, 0, 2, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2,
       0, 2, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 3, 0, 3, 1, 1, 1, 1, 1, 0, 1, 3,
       1, 3, 1, 1, 1, 3, 0, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 3, 1, 1, 1, 1,
       3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 1, 3, 3, 1, 3, 1, 1, 3, 3,
       3, 3, 1, 1, 1, 1, 3, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 3, 1, 3, 3, 1,
       3, 1, 1, 3, 1, 1, 3, 3, 3, 1, 1, 1, 1, 3, 1, 3, 3, 3, 1, 1, 1, 3,
       3, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [103…
```
plt.figure(figsize=(10,8))
plt.scatter(data2['Country'], data2['Birthrate'], c=agg_cluster.labels_)
```

Out[103]:
```
<matplotlib.collections.PathCollection at 0x23f730cd1c0>
```



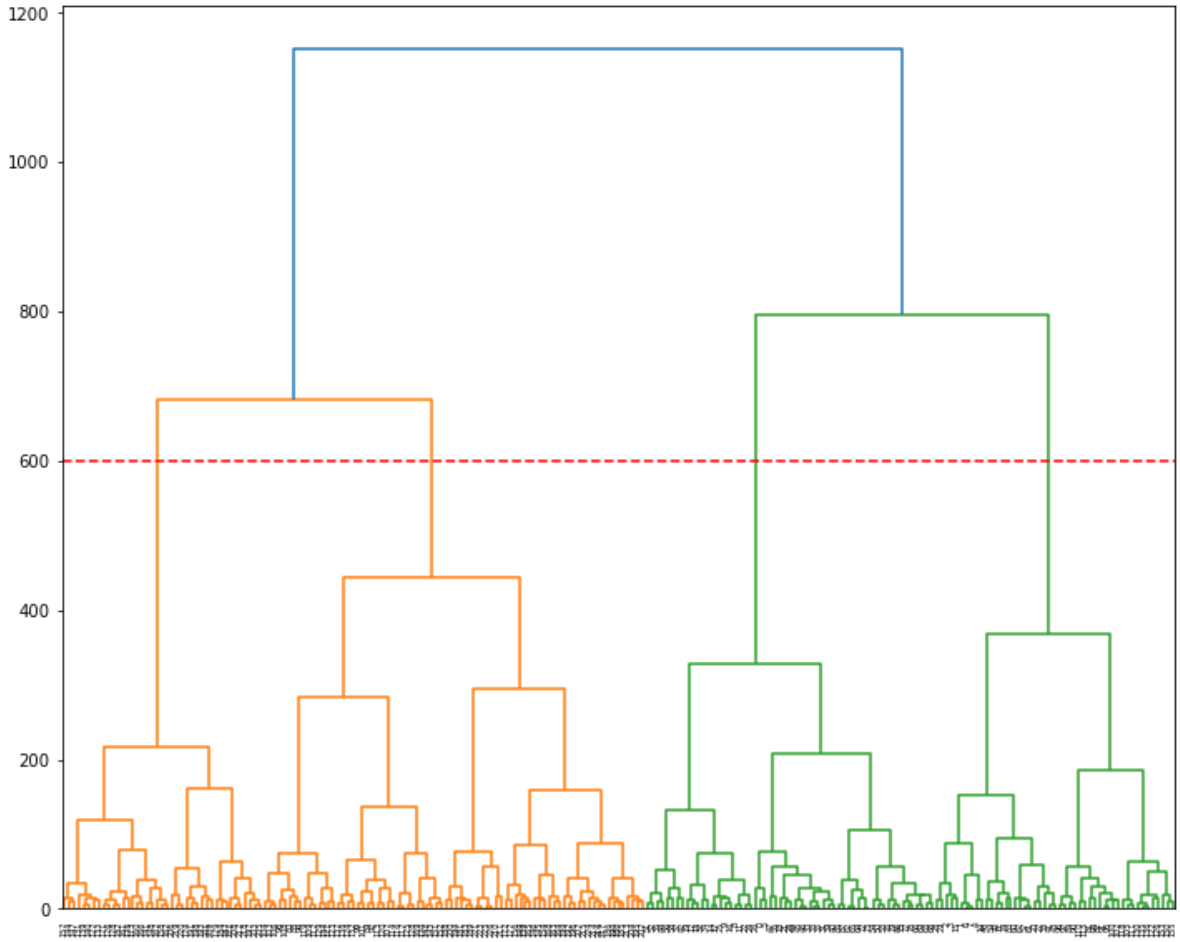
3.3 Clustering Countries on Deathrate

In [116…
```
data3=data1[['Country', 'Region', 'Deathrate']]
data3.head()
```

Out[116]:

	Country	Region	Deathrate
0	0	0	190
1	1	3	35
2	2	6	21
3	3	8	4
4	4	10	61

In [118…
```
plt.figure(figsize=(12,10))
hc.dendrogram(hc.linkage(data3, method='ward'))
plt.axhline(y=600, color='r', linestyle='--')
plt.show();
```

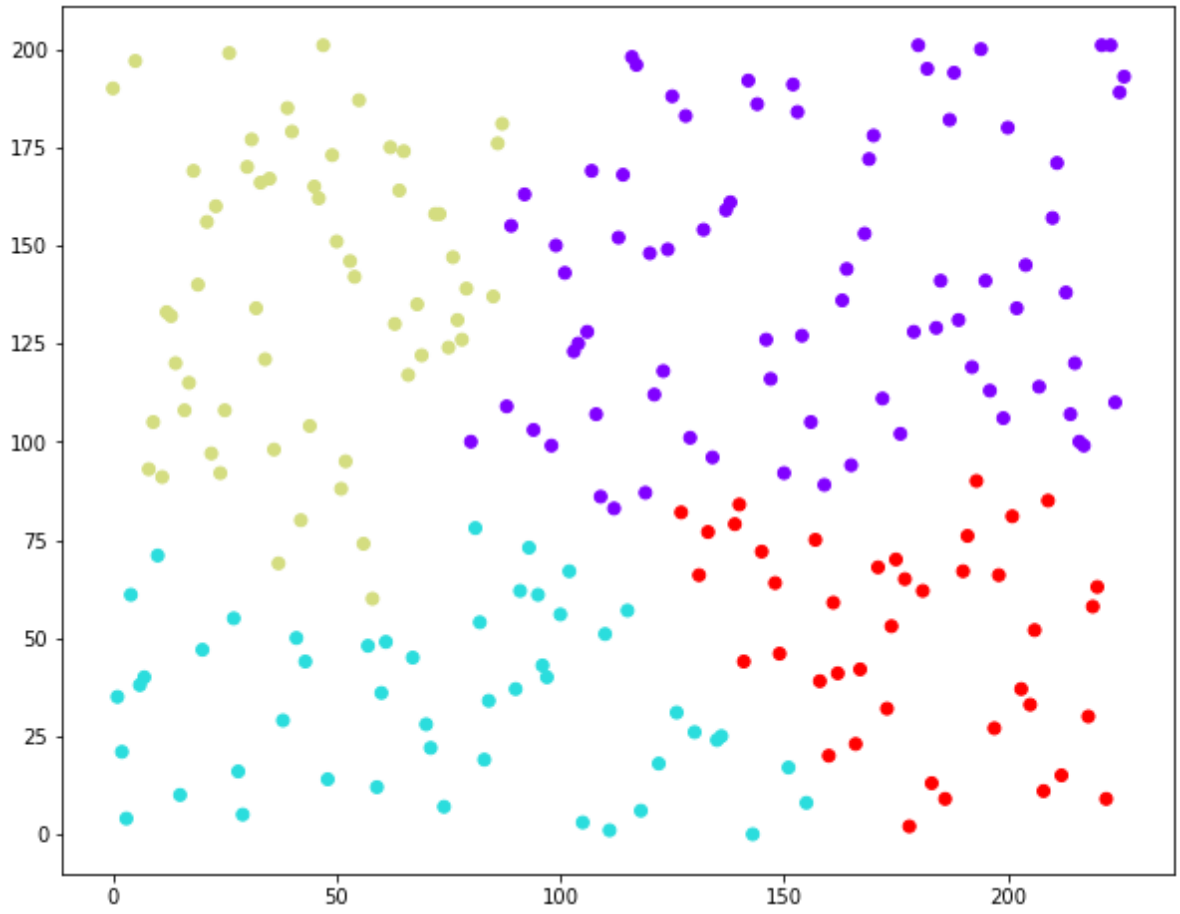


In [119…
```
agg_cluster= AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='
labels_= agg_cluster.fit_predict(data3)
labels_
```

Out[119]:
```
array([2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2,
       2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 1,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2,
       2, 1, 2, 2, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 0, 1, 1, 1, 1, 2, 2, 2,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 3, 0, 0, 1, 3,
       0, 3, 0, 1, 1, 0, 0, 3, 3, 3, 0, 0, 3, 0, 0, 3, 3, 0, 1, 0, 0, 1,
       0, 1, 0, 3, 3, 0, 3, 3, 3, 0, 0, 0, 3, 3, 0, 0, 0, 3, 0, 3, 3, 3,
       3, 0, 3, 3, 0, 0, 3, 0, 3, 0, 0, 3, 0, 0, 0, 3, 0, 3, 0, 0, 0, 3,
       3, 0, 0, 3, 0, 3, 0, 3, 3, 0, 3, 3, 0, 0, 3, 0, 0, 0, 0, 0, 3, 3,
       3, 0, 3, 0, 0, 0, 0], dtype=int64)
```

In [122…
```
plt.figure(figsize=(10,8))
plt.scatter(data3['Country'], data3['Deathrate'], c=agg_cluster.labels_, cmap='rai
```

Out[122]:
```
<matplotlib.collections.PathCollection at 0x23f7758cfa0>
```



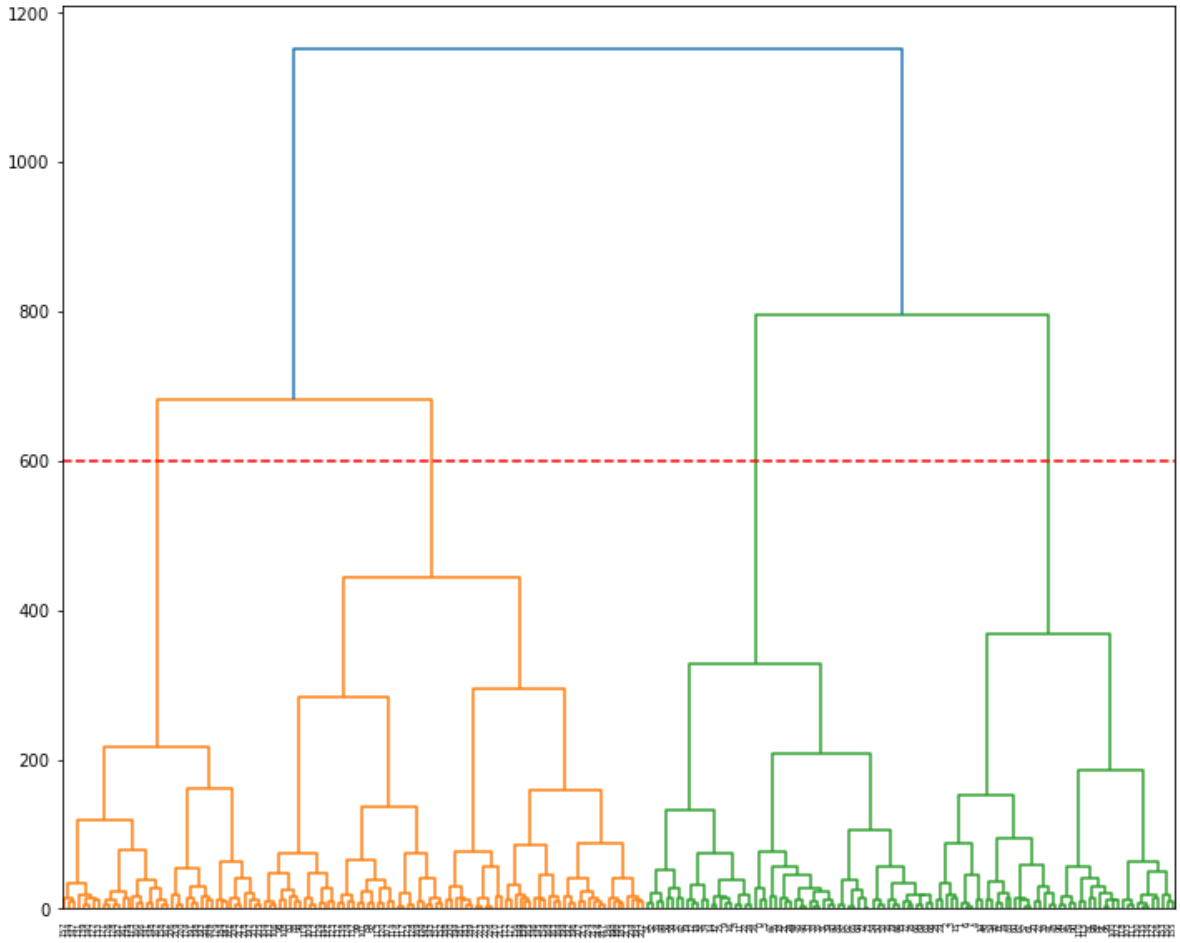
3.4 Clustering Countries on Birthrate and Deathrate

In [124…
```
data4=data1[['Country', 'Birthrate', 'Deathrate']]
data4.head()
```

Out[124]:

	Country	Birthrate	Deathrate
0	0	216	190
1	1	74	35
2	2	95	21
3	3	131	4
4	4	3	61

In [126…
```
plt.figure(figsize=(12,10))
hc.dendrogram(hc.linkage(data3, method='ward'))
plt.axhline(y=600, color='r', linestyle='--')
plt.show();
```



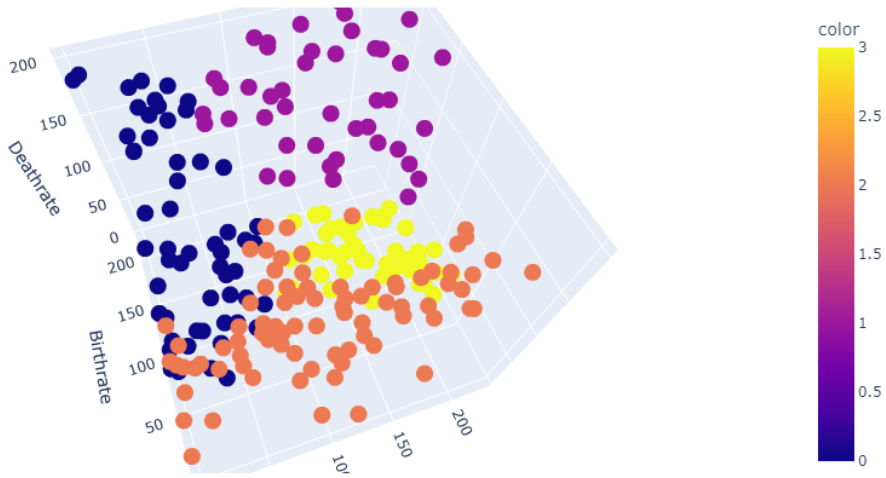
In [127…
```
agg_cluster= AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='
labels_= agg_cluster.fit_predict(data4)
labels_
```

Out[127]:
```
array([0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 0, 0,
       2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       2, 0, 2, 2, 0, 0, 1, 1, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 2, 1, 1,
       2, 1, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 3, 2, 1, 1, 2,
       2, 3, 2, 1, 2, 2, 1, 1, 3, 2, 2, 2, 2, 2, 1, 1, 3, 1, 1, 2, 3, 2,
       1, 2, 1, 3, 3, 2, 2, 3, 3, 3, 1, 1, 3, 1, 2, 2, 3, 2, 3, 1, 1, 3,
       2, 3, 1, 3, 3, 1, 3, 3, 3, 2, 2, 2, 3, 3, 2, 2, 1, 2, 3, 3, 3, 3,
       3, 2, 1, 3, 1, 1, 3, 1, 2, 2, 2, 3, 1, 1, 2, 3, 1, 3, 1, 2, 2, 3,
       3, 1, 1, 3, 1, 3, 2, 3, 3, 1, 3, 1, 1, 2, 3, 2, 2, 2, 1, 1, 3, 3,
       3, 1, 3, 1, 1, 1, 1], dtype=int64)
```

In [131…
```
px.scatter_3d(data_frame=data4, x='Country', y='Birthrate', z='Deathrate', color=ag
```

In [132…
```
from IPython import display
display.Image("image1.png")
```

Out[132]:



In [ ]: