K Means Clustering

Data: USArrests

Libraries

```
# God is Great!
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
pd.set_option('display.max_column',None)
from random import sample
from sklearn import preprocessing
from sklearn.preprocessing import scale
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from scipy.spatial import distance_matrix
from scipy.spatial import distance
from sklearn import datasets
from sklearn.metrics import silhouette_score
from sklearn.metrics import silhouette_samples
from sklearn.decomposition import PCA
```

Data

```
In [2]: data=pd.read_csv("C:/Users/Dr Vinod/Desktop/USArrests.csv")

In [3]: data=pd.DataFrame(data)

In [4]: data.head()
Out[4]:
        State  Murder  Assault  UrbanPop  Rape
0     Alabama    13.2      236        58  21.2
1      Alaska    10.0      263        48  44.5
2     Arizona     8.1      294        80  31.0
3    Arkansas     8.8      190        50  19.5
4  California     9.0      276        91  40.6

In [5]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   State     50 non-null     object
 1   Murder    50 non-null     float64
 2   Assault   50 non-null     int64
 3   UrbanPop  50 non-null     int64
 4   Rape      50 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 2.1+ KB
```

Set Index as 'state'

```
# Making the column "State" as the index
data=data.set_index('State')
data.head()
```

```
In [7]: data.head()
Out[7]:
            Murder  Assault  UrbanPop  Rape
State
Alabama       13.2      236        58  21.2
Alaska        10.0      263        48  44.5
Arizona        8.1      294        80  31.0
Arkansas       8.8      190        50  19.5
California     9.0      276        91  40.6
```

data - DataFrame

State	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	263	48	44.5
Arizona	8.1	294	80	31
Arkansas	8.8	190	50	19.5
California	9	276	91	40.6
Colorado	7.9	204	78	38.7

Sample
10 states

```
# Creating a sample of 10 rows from the USArrests dataset
sample=data.sample(frac=0.20,replace=False,random_state=123)
len(sample)
sample
```

```
In [10]: sample
Out[10]:
```

State	Murder	Assault	UrbanPop	Rape
Hawaii	5.3	46	83	20.2
Indiana	7.2	113	65	21.0
New Mexico	11.4	285	70	32.1
Washington	4.0	145	73	26.2
Maine	2.1	83	51	7.8
Alabama	13.2	236	58	21.2
South Dakota	3.8	86	45	12.8
Illinois	10.4	249	83	24.0
New Jersey	7.4	159	89	18.8
Florida	15.4	335	80	31.9

Distance Matrix to excel

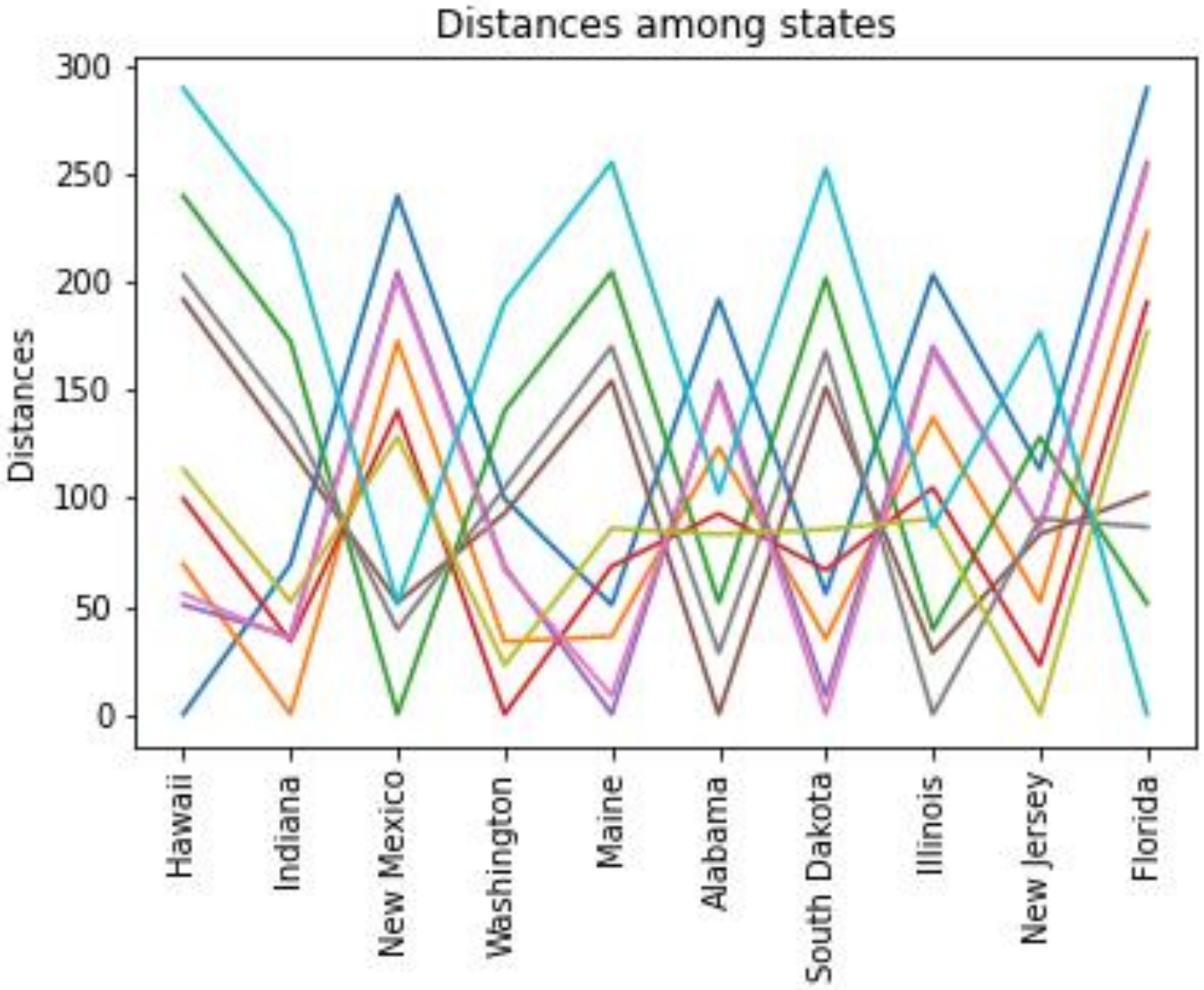
Index means 1st column

```
# Computing Distance Matrix
DM=pd.DataFrame(distance_matrix(sample.values, sample.values), index=sample.index,
                columns=sample.index)
round(DM, 2)
DM.to_csv('DM2.csv')
```

	A	B	C	D	E	F	G	H	I	J	K
1	State	Hawaii	Indiana	New Mexi	Washingto	Maine	Alabama	South Dak	Illinois	New Jerse	Florida
2	Hawaii	0	69.40641	239.7266	99.69298	50.56679	191.8031	55.68671	203.0996	113.1873	289.4286
3	Indiana	69.40641	0	172.4814	33.54519	36.00347	123.3452	34.75342	137.2561	51.93149	222.9239
4	New Mexico	239.7266	172.4814	0	140.3516	204.5531	51.64349	201.634	39.13579	128.1791	51.14724
5	Washington	99.69298	33.54519	140.3516	0	68.33864	92.82047	66.66783	104.6986	22.76664	190.5556
6	Maine	50.56679	36.00347	204.5531	68.33864	0	154.1453	8.537564	170.0333	85.8434	255.1523
7	Alabama	191.8031	123.3452	51.64349	92.82047	154.1453	0	151.0891	28.45488	83.24302	102.0016
8	South Dakota	55.68671	34.75342	201.634	66.66783	8.537564	151.0891	0	167.875	85.52169	252.4388
9	Illinois	203.0996	137.2561	39.13579	104.6986	170.0333	28.45488	167.875	0	90.39934	86.55871
10	New Jersey	113.1873	51.93149	128.1791	22.76664	85.8434	83.24302	85.52169	90.39934	0	176.8972
11	Florida	289.4286	222.9239	51.14724	190.5556	255.1523	102.0016	252.4388	86.55871	176.8972	0

Visualize the matrix

```
# Visualizing the distance using matplotlib
plt.plot(DM)
plt.ylabel("Distances")
plt.grid(False)
plt.xticks(rotation = 90)
plt.title('Distances among states')
plt.show()
```



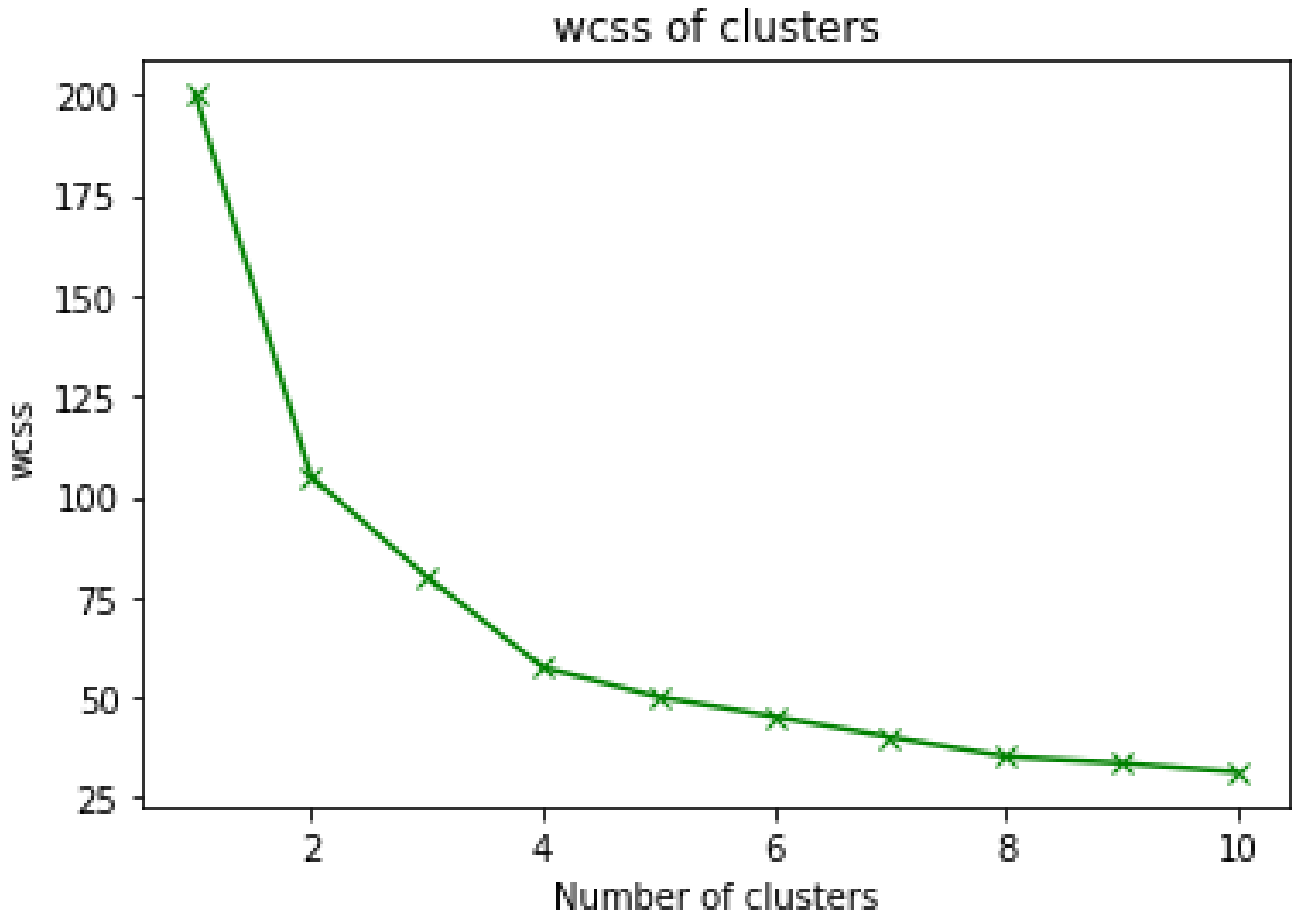
Scale the data

```
data_scaled=StandardScaler().fit_transform(data)
data_scaled
```

```
In [19]: data_scaled
Out[19]:
array([[ 1.25517927,  0.79078716, -0.52619514, -0.00345116],
       [ 0.51301858,  1.11805959, -1.22406668,  2.50942392],
       [ 0.07236067,  1.49381682,  1.00912225,  1.05346626],
       [ 0.23470832,  0.23321191, -1.08449238, -0.18679398],
       [ 0.28109336,  1.2756352 ,  1.77678094,  2.08881393],
       [ 0.02597562,  0.40290872,  0.86954794,  1.88390137],
       [-1.04088037, -0.73648418,  0.79976079, -1.09272319],
       [-0.43787481,  0.81502956,  0.45082502, -0.58583422],
       [ 1.76541475,  1.99078607,  1.00912225,  1.1505301 ],
       [ 2.22926518,  0.48775713, -0.38662083,  0.49265293],
       [ 0.16513075, -0.17890893, -0.17725937, -0.05737552],
       [-0.87853272, -0.31224214,  0.52061217,  0.53579242],
       [-0.48425985, -1.08799901, -1.85215107, -1.28685088],
       [-1.20322802, -1.42739264,  0.03210209, -1.1250778 ],
       [-0.22914211, -0.11830292, -0.38662083, -0.60740397]])
```

Find optimum nos of clusters

```
plt.figure(figsize = (10,8))

wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters= i, init= 'random', random_state= 42)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)

wcss #10 values will appear

plt.plot(range(1,11), wcss, 'gx-')
plt.title('wcss of clusters')
plt.xlabel('Number of clusters')
plt.ylabel('wcss')
plt.show()
```


wcss of clusters
200
175
150
125
100
75
50
25
wcss
2
4
6
8
10
Number of clusters

Clustering happened!

```
# Running KMeans to our desired number/optimum no of clusters (k = 4)
# just making the cluster in the backend (not fitted to dataset here)
kmeans=KMeans(n_clusters=4)
# fitting the cluster to the dataset
clusters=kmeans.fit_predict(data_scaled)
# cluster allocation
clusters
```

```
In [22]: clusters
Out[22]:
array([1, 0, 0, 1, 0, 0, 3, 3, 0, 1, 3, 2, 0, 3, 2, 3, 2, 1, 2, 0, 3, 0,
       2, 1, 0, 2, 2, 0, 2, 3, 0, 0, 1, 2, 3, 3, 3, 3, 3, 1, 2, 1, 0, 3,
       2, 3, 3, 2, 2, 3])
```

Cluster Membership

```
# We are not comfortable with the cluster 0, so 1 is added to the clusters
# to change cluster 0 to cluster 1, 1 to 2, 2 to 3 and so on.
# Therefore naming clusters 1-4 instead of 0-3 and adding to the dataframe
Final_Clusters=clusters+1
cluster=list(Final_Clusters)
data['member']=cluster # Addition of the column to the dataset
data.head()
```

```
In [26]: data.head()
Out[26]:
             Murder  Assault  UrbanPop  Rape  member
State
Alabama        13.2      236        58  21.2       2
Alaska         10.0      263        48  44.5       1
Arizona         8.1      294        80  31.0       1
Arkansas        8.8      190        50  19.5       2
California      9.0      276        91  40.6       1
```

Overall Silhouette Score

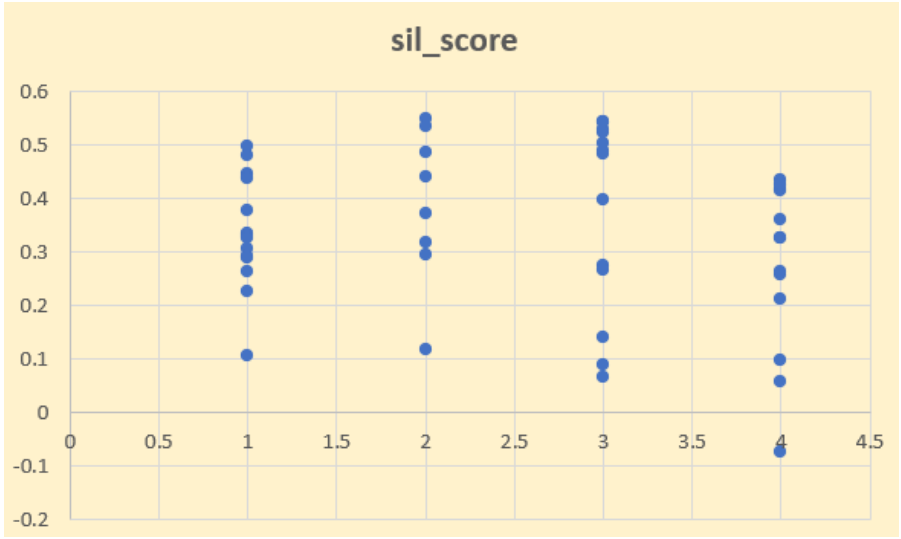
```
#__________silhouette score_OVERALL
print(f'sil score(n=4): {silhouette_score(data_scaled, cluster)}') #0.34
```

```
In [36]: print(f'sil score(n=4): {silhouette_score(data_scaled, cluster)}') #0.34
sil score(n=4): 0.33968891433344395
```

Silhouette Score of each data point

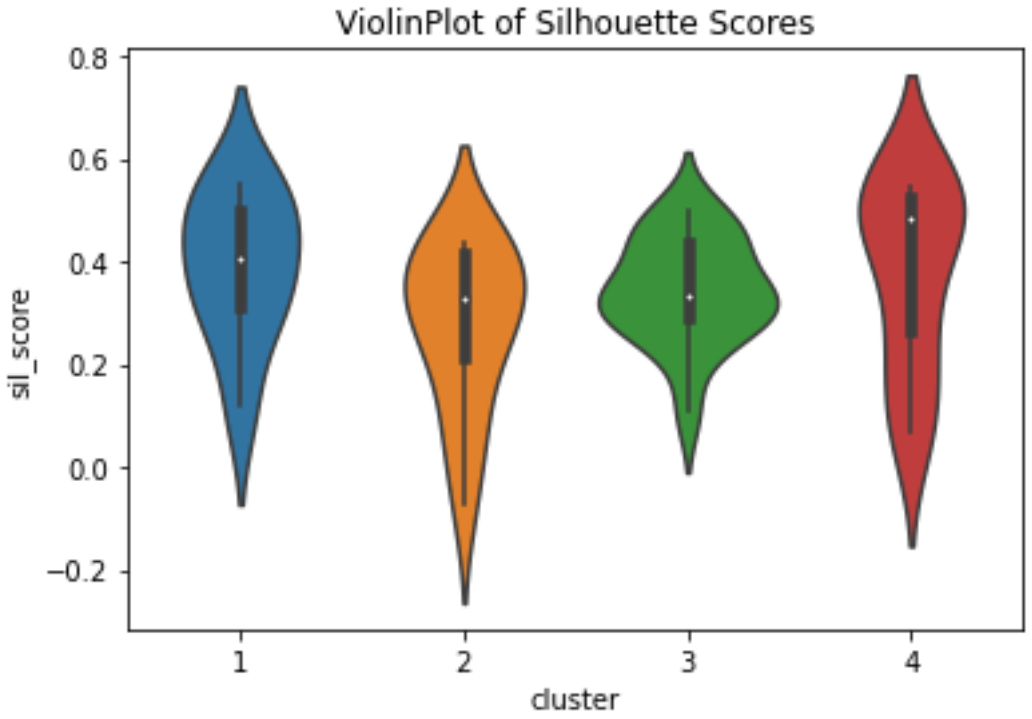
```
#_____silhouette score--> EACH DATA POINT
sample_silhouette_values = silhouette_samples(data_scaled, cluster)

# sample_silhouette_values is an array, convert to DF
sample_silhouette_values = pd.DataFrame(sample_silhouette_values)
# Export to desktop
sample_silhouette_values.to_csv('C:/Users/Dr Vinod/Desktop/sil_values4.csv')
```

	A	B	C	D	E	F	G
1	State	Murder	Assault	UrbanPop	Rape	cluster	sil_score
2	Alabama	13.2	236	58	21.2	2	0.485775
3	Alaska	10	263	48	44.5	4	0.058252
4	Arizona	8.1	294	80	31	4	0.415483
5	Arkansas	8.8	190	50	19.5	2	0.118709
6	California	9	276	91	40.6	4	0.435559
7	Colorado	7.9	204	78	38.7	4	0.326542
8	Connecticu	3.3	110	77	11.1	1	0.227173
9	Delaware	5.9	238	72	15.8	1	0.332977
10	Florida	15.4	335	80	31.9	4	0.258017
11	Georgia	17.4	211	60	25.8	2	0.371803

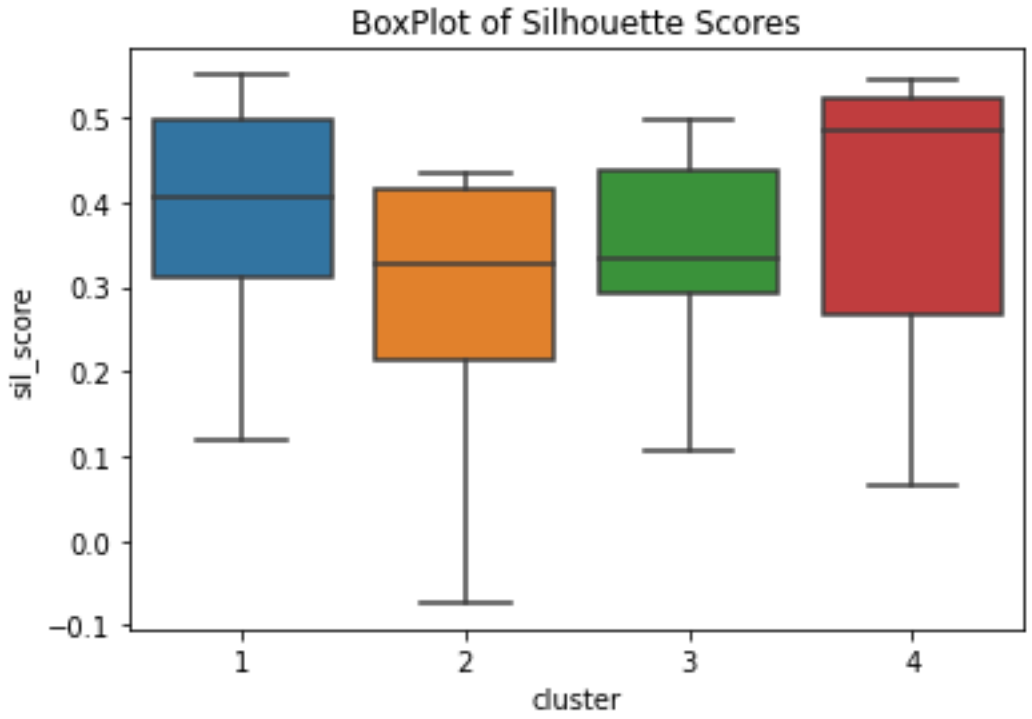


Pattern of sil_score across clusters

```
sns.violinplot(x = 'cluster', y = 'sil_score', data = data1)
plt.title('ViolinPlot of Silhouette Scores')
```




```
sns.boxplot(x = 'cluster', y = 'sil_score', data = data1)
plt.title('BoxPlot of Silhouette Scores')
```




```
#_____________________________________complete data
data.info() # cluster added already
data['sil_score'] = sample_silhouette_values # add silscore
data.to_csv('C:/Users/Dr Vinod/Desktop/ClstrUSArrsts.csv')
data1=pd.read_csv("C:/Users/Dr Vinod/Desktop/ClstrUSArrsts.csv")
data1=pd.DataFrame(data1)
```

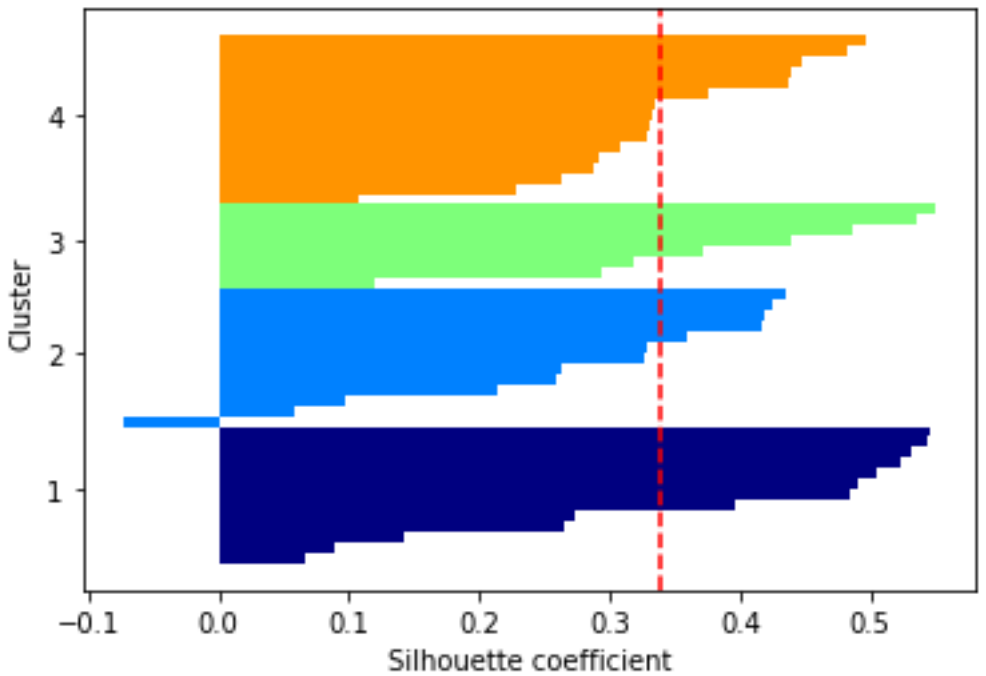
Cluster wise Sil_score

```
#_____silhouette score of each cluster
from sklearn.metrics import silhouette_samples

num_clusters = 4
# sil score of each data point
sample_silhouette_values = silhouette_samples(data_scaled, cluster)

means_list = [] #empty list
for i in range(num_clusters):
    means_list.append(round(sample_silhouette_values[i == clusters].mean(),2))
    # clusters=kmeans.fit_predict(data_scaled); 0,1,2,3
print(means_list)  # 0.27, 0.39, 0.37, 0.34
```

Do it again as we had made a data frame of 'sample_silhouette_values' before

```
In [107]: means_list = [] #empty list
     ...: for i in range(num_clusters):
     ...:     means_list.append(round(sample_silhouette_values[i == clusters].mean(),2))
     ...:     # clusters=kmeans.fit_predict(data_scaled); 0,1,2,3
     ...: print(means_list)  # 0.27, 0.39, 0.37, 0.34
[0.27, 0.39, 0.37, 0.34]
```

```
#_____plotting sil score
import numpy as np
from matplotlib import cm
from sklearn.metrics import silhouette_samples

cluster_labels = np.unique(clusters)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(data_scaled,clusters,metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []

for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[clusters == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper),
             c_silhouette_vals,
             height=1.0,
             edgecolor='none',
             color=color)
    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)
silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg,
            color="red",
            linestyle="--")
plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')
```

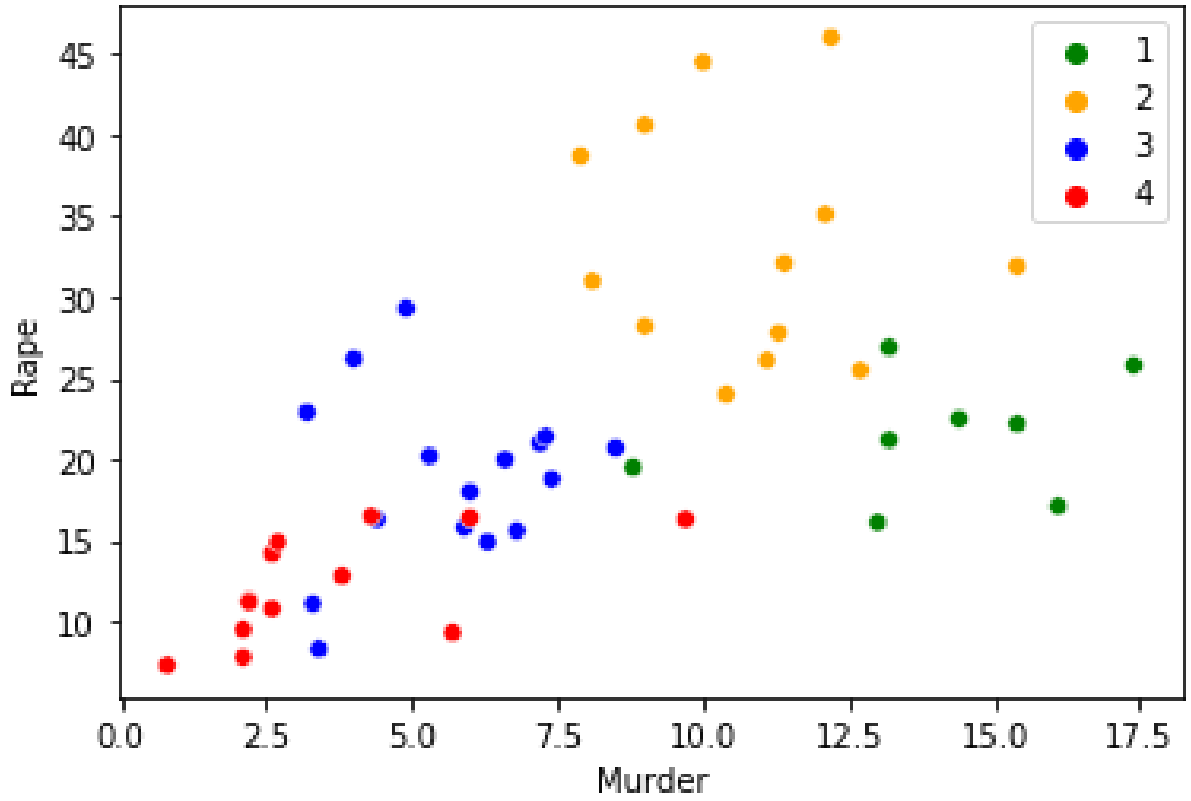


Profiling of clusters

```
#____________cluster profile
data.groupby(data['cluster']).mean()
```

```
In [35]: data.groupby(data['cluster']).mean()
Out[35]:
              Murder     Assault   UrbanPop       Rape
cluster
1           3.600000   78.538462  52.076923  12.176923
2          10.815385  257.384615  76.000000  33.192308
3          13.937500  243.625000  53.750000  21.412500
4           5.656250  138.875000  73.875000  18.781250
```

```
#_______cluster plot
# plotting the scatterplot

plt.figure(figsize=(12,6))

sns.scatterplot(data['Murder'],data['Rape'],hue=Final_Clusters,
                palette=['green','orange','blue','red'])
```




A LEARNING CURVE
IS ESSENTIAL TO
Growth
LEARNING
IN BLOOM