Entrée [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics
from sklearn.cluster import KMeans
import sklearn.cluster as cluster
import warnings
warnings.filterwarnings('ignore')
```

Entrée [2]:

```
l1= pd.Series({'Attribute1':2,'Attribute2':10})
l2= pd.Series({'Attribute1':2,'Attribute2':5})
l3= pd.Series({'Attribute1':8,'Attribute2':4})
l4= pd.Series({'Attribute1':5,'Attribute2':8})
l5= pd.Series({'Attribute1':7,'Attribute2':5})
l6= pd.Series({'Attribute1':6,'Attribute2':4})
l7= pd.Series({'Attribute1':1,'Attribute2':2})
l8= pd.Series({'Attribute1':4,'Attribute2':9})
df = pd.DataFrame([l1,l2,l3,l4,l5,l6,l7,l8])
df
```

Out[2]:

	Attribute1	Attribute2
0	2	10
1	2	5
2	8	4
3	5	8
4	7	5
5	6	4
6	1	2
7	4	9

Entrée [3]:

```
scaler = MinMaxScaler()
scale=scaler.fit_transform(df[['Attribute1','Attribute2']])
df_scale=pd.DataFrame(scale,columns=['Attribute1','Attribute2']);
df_scale
```

Out[3]:

	Attribute1	Attribute2
0	0.142857	1.000
1	0.142857	0.375
2	1.000000	0.250
3	0.571429	0.750
4	0.857143	0.375
5	0.714286	0.250
6	0.000000	0.000
7	0.428571	0.875

Entrée [4]:

```
df_scale.shape
```

Out[4]:

```
(8, 2)
```

Entrée [5]:

```
df_scale.head(4)
```

Out[5]:

	Attribute1	Attribute2
0	0.142857	1.000
1	0.142857	0.375
2	1.000000	0.250
3	0.571429	0.750

Entrée [6]:

```
df_scale.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Attribute1  8 non-null      float64
 1   Attribute2  8 non-null      float64
dtypes: float64(2)
memory usage: 256.0 bytes
```

Entrée [7]:

```
df_scale.isnull().sum()
```

Out[7]:

```
Attribute1    0
Attribute2    0
dtype: int64
```

Entrée [8]:

```
df_scale.describe()
```

Out[8]:

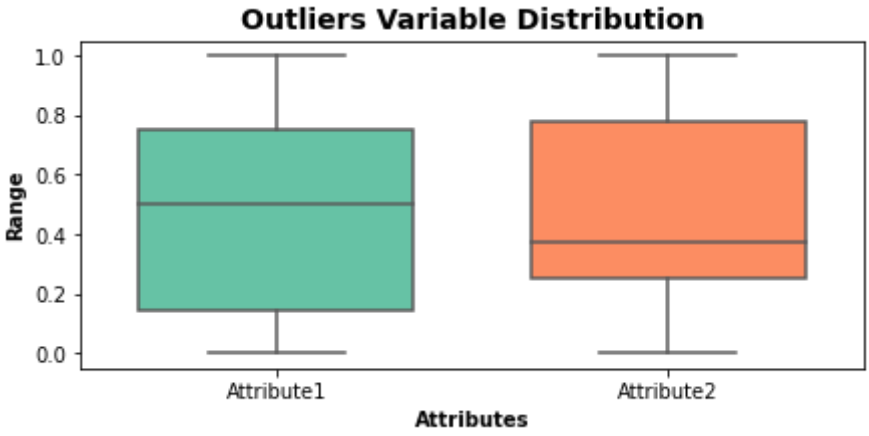
	Attribute1	Attribute2
count	8.000000	8.000000
mean	0.482143	0.484375
std	0.365713	0.349984
min	0.000000	0.000000
25%	0.142857	0.250000
50%	0.500000	0.375000
75%	0.750000	0.781250
max	1.000000	1.000000

Entrée [9]:

```
attributes = ['Attribute1' , 'Attribute2']
plt.rcParams['figure.figsize'] = [7,3]
sns.boxplot(data = df_scale[attributes], orient="v", palette="Set2" ,whis=1.5,saturation=1,
plt.title("Outliers Variable Distribution", fontsize = 14, fontweight = 'bold')
plt.ylabel("Range", fontweight = 'bold')
plt.xlabel("Attributes", fontweight = 'bold')
```

Out[9]:

```
Text(0.5, 0, 'Attributes')
```



Entrée [10]:

```
km=KMeans(n_clusters=3)
```

Entrée [11]:

```
y_km = km.fit_predict(df[['Attribute1','Attribute2']])
y_km
```

Out[11]:

```
array([1, 2, 0, 1, 0, 0, 2, 1])
```

Entrée [12]:

```
km.cluster_centers_
```

Out[12]:

```
array([[7.        , 4.33333333],
       [3.66666667, 9.        ],
       [1.5       , 3.5       ]])
```

Entrée [13]:

```
km.cluster_centers_.reshape(2,3)
```

Out[13]:

```
array([[7.        , 4.33333333, 3.66666667],
       [9.        , 1.5       , 3.5       ]])
```

Entrée [14]:

```
kmeans=cluster.KMeans(n_clusters=3,init="k-means++")
kmeans = kmeans.fit(df[['Attribute1','Attribute2']])
```

Entrée [15]:

```
y_kmeans = kmeans.fit_predict(df[['Attribute1','Attribute2']])
y_kmeans
```

Out[15]:

```
array([0, 2, 1, 0, 1, 1, 2, 0])
```

Entrée [16]:

```
kmeans.cluster_centers_
```

Out[16]:

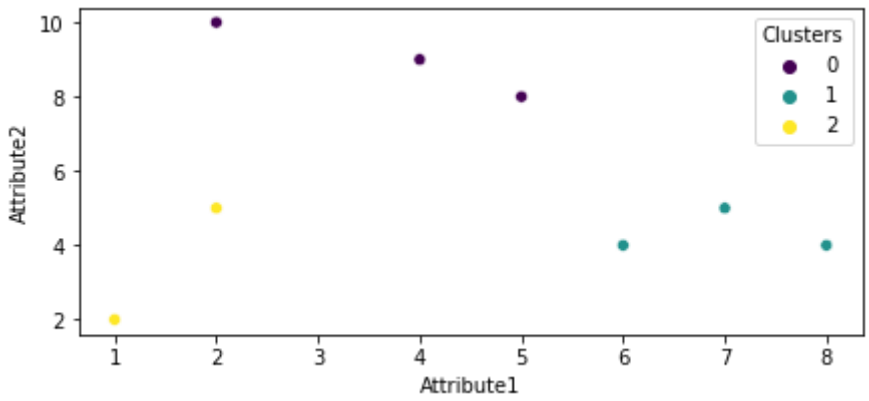
```
array([[3.66666667, 9.        ],
       [7.        , 4.33333333],
       [1.5       , 3.5       ]])
```

Entrée [17]:

```
df["Clusters"] = kmeans.labels_
sns.scatterplot(x="Attribute1",y="Attribute2",hue="Clusters",data=df,palette='viridis')
```

Out[17]:

```
<AxesSubplot:xlabel='Attribute1', ylabel='Attribute2'>
```



Entrée [18]:

```
K = range(2,8)
wss=[]
```

Entrée [19]:

```
for k in K:
    kmeans=cluster.KMeans(n_clusters=k)
```

Entrée [20]:

```
kmeans=kmeans.fit(df_scale)
```

Entrée [21]:

```
wss_iter=kmeans.inertia_
wss.append(wss_iter)
```

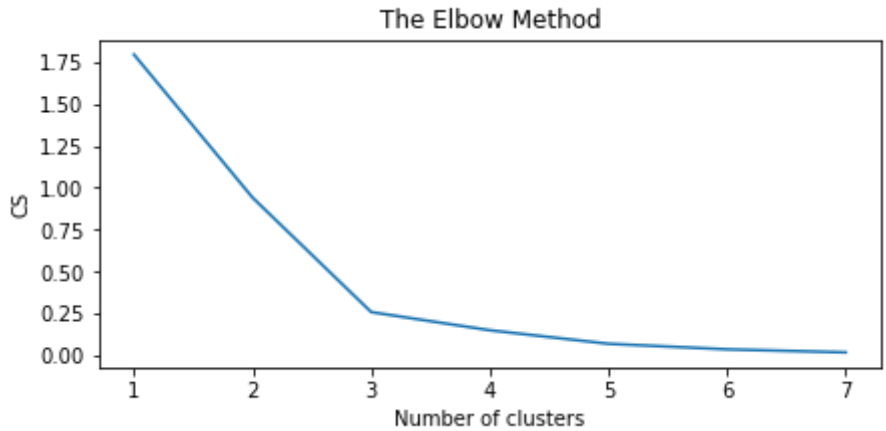
Entrée [22]:

```
import sklearn.cluster as cluster
import sklearn.metrics as metrics
for i in range(2,8):
    labels=cluster.KMeans(n_clusters=i,random_state=200).fit(df_scale).labels_
    print ("Silhouette score for k(clusters) = "+str(i)+" is "
    +str(metrics.silhouette_score(df_scale,labels,metric="euclidean",sample_size=1000,rando
```

```
Silhouette score for k(clusters) = 2 is 0.3960790641612191
Silhouette score for k(clusters) = 3 is 0.5648525330501148
Silhouette score for k(clusters) = 4 is 0.47751663091430685
Silhouette score for k(clusters) = 5 is 0.3687975125416946
Silhouette score for k(clusters) = 6 is 0.16804352358468255
Silhouette score for k(clusters) = 7 is 0.04195198583801057
```

Entrée [23]:

```
from sklearn.cluster import KMeans
cs = []
for i in range(1, 8):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random
    kmeans.fit(df_scale)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 8), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```



So the best K value is 3 as it has highest score.