```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
crime_data= pd.read_csv('/content/drive/MyDrive/Personal/Studies/MSC Data Science Material/SEM2/ML/Practical/data_set/crime_data.csv')
crime_data
```

18	Maine	2.1	83	51	7.8
19	Maryland	11.3	300	67	27.8
20	Massachusetts	4.4	149	85	16.3
21	Michigan	12.1	255	74	35.1
22	Minnesota	2.7	72	66	14.9
23	Mississippi	16.1	259	44	17.1
24	Missouri	9.0	178	70	28.2
25	Montana	6.0	109	53	16.4
26	Nebraska	4.3	102	62	16.5
27	Nevada	12.2	252	81	46.0
28	New Hampshire	2.1	57	56	9.5
29	New Jersey	7.4	159	89	18.8

```
crime_data.isnull().any()
```

```
Unnamed: 0    False
Murder        False
Assault       False
UrbanPop      False
Rape          False
dtype: bool
```

35	Oklahoma	6.6	151	68	20.0

```
mydata=crime_data.iloc[:,crime_data.columns!='Unnamed: 0']
inplace=True
```

```
crime_data.head()
```

	Unnamed: 0	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

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Murder    50 non-null     float64
 1   Assault   50 non-null     int64  
 2   UrbanPop  50 non-null     int64  
 3   Rape      50 non-null     float64
dtypes: float64(2), int64(2)
memory usage: 1.7 KB
```

```
scaler=MinMaxScaler()
norm_mydata=mydata.copy()
```

```
def minmaxscaler(x):
  for columnName, columnData in x.iteritems():
    x[columnName]=scaler.fit_transform(np.array(columnData).reshape(-1,1)) #Decimal method
```

```
minmaxscaler(norm_mydata)
```

```
<ipython-input-10-8f583cea87c9>:2: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items inste
  for columnName, columnData in x.iteritems():
```

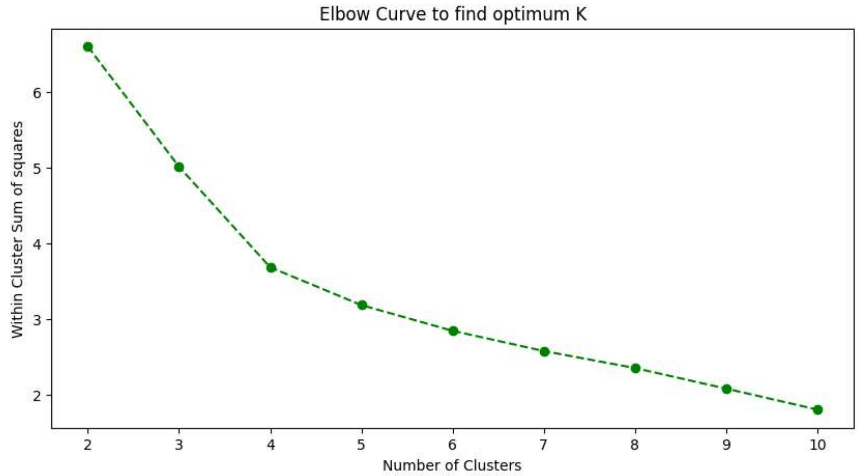
```
norm_mydata
```

18	0.078313	0.130137	0.322034	0.012920
19	0.632530	0.873288	0.593220	0.529716
20	0.216867	0.356164	0.898305	0.232558
21	0.680723	0.719178	0.711864	0.718346
22	0.114458	0.092466	0.576271	0.196382
23	0.921687	0.732877	0.203390	0.253230
24	0.493976	0.455479	0.644068	0.540052
25	0.313253	0.219178	0.355932	0.235142
26	0.210843	0.195205	0.508475	0.237726
27	0.686747	0.708904	0.830508	1.000000
28	0.078313	0.041096	0.406780	0.056848
29	0.397590	0.390411	0.966102	0.297158
30	0.638554	0.821918	0.644068	0.640827
31	0.620482	0.715753	0.915254	0.485788
32	0.734940	1.000000	0.220339	0.227390
33	0.000000	0.000000	0.203390	0.000000
34	0.391566	0.256849	0.728814	0.364341
35	0.349398	0.363014	0.610169	0.328165
36	0.246988	0.390411	0.593220	0.568475
37	0.331325	0.208904	0.677966	0.196382
38	0.156627	0.441781	0.932203	0.025840
39	0.819277	0.801370	0.271186	0.392765
40	0.180723	0.140411	0.220339	0.142119
41	0.746988	0.489726	0.457627	0.506460
42	0.716867	0.534247	0.813559	0.470284
43	0.144578	0.256849	0.813559	0.403101
44	0.084337	0.010274	0.000000	0.100775
45	0.463855	0.380137	0.525424	0.346253
46	0.192771	0.342466	0.694915	0.488372
47	0.295181	0.123288	0.118644	0.051680
48	0.108434	0.027397	0.576271	0.090439
49	0.361446	0.397260	0.474576	0.214470

```
#Scree plot or Elbow plot to find k
```

```
k=list(range(2,11))
sum_of_squared_distances = []

for i in k:
  kmeans=KMeans(n_clusters=i)
  kmeans.fit(norm_mydata)
  sum_of_squared_distances.append(kmeans.inertia_)

plt.figure(figsize=(10,5))
plt.plot(k, sum_of_squared_distances, 'go--')
plt.xlabel('Number of Clusters')
plt.ylabel('Within Cluster Sum of squares')
plt.title('Elbow Curve to find optimum K')
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning
  warnings.warn(
Text(0.5, 1.0, 'Elbow Curve to find optimum K')
```




```
#Now building kmeans model with k=4

#Instantiating
kmeans4=KMeans(n_clusters=4)

#Training the model
kmeans4.fit(norm_mydata)

#predicting
y_pred=kmeans4.fit_predict(norm_mydata)
print(y_pred)

#Storing the y_pred values in a new column
crime_data['Cluster']=y_pred+1 #to start the cluster number from 1
```

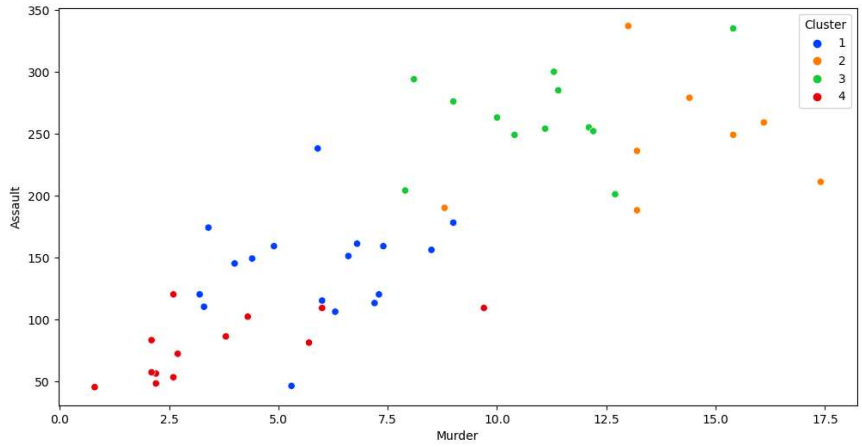
```
[1 2 2 1 2 2 0 0 2 1 0 3 2 0 3 0 3 1 3 2 0 2 3 1 0 3 3 2 3 0 2 2 1 3 0 0 0
 0 0 1 3 1 2 0 3 0 0 3 3 0]
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
```

```
#Storing the centroids to a dataframe

centroids=kmeans4.cluster_centers_
centroids=pd.DataFrame(centroids, columns=['Murder','Assault','UrbanPop','Rape'])
centroids.index=np.arange(1,len(centroids)+1) #Start the index from 1
centroids
```

	Murder	Assault	UrbanPop	Rape
1	0.304394	0.329371	0.705882	0.310990
2	0.791416	0.680223	0.368644	0.364664
3	0.612450	0.750000	0.754237	0.679802
4	0.168675	0.114858	0.340287	0.126019

```
#Sample visualization of clusters
#Lets just take any two of the features and plot to see how the observations are clustered

import seaborn as sns
plt.figure(figsize=(12,6))
sns.set_palette("pastel")
sns.scatterplot(x=crime_data['Murder'], y=crime_data['Assault'], hue=crime_data['Cluster'], palette='bright')
```

```
<Axes: xlabel='Murder', ylabel='Assault'>
```




```
sns.scatterplot(x=crime_data['Rape'], y=crime_data['UrbanPop'], hue=crime_data['Cluster'], palette='bright')
```

```
<Axes: xlabel='Rape', ylabel='UrbanPop'>
```