

My Assumptions:

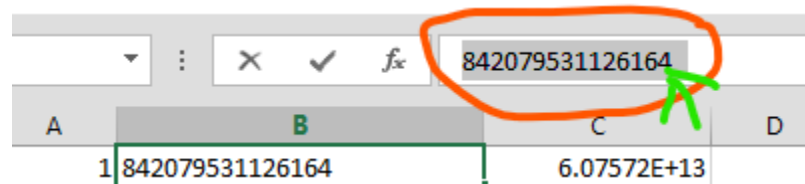
1. As according to the given instruction k-means.csv should be consist of 2 columns (x1, x2) but actually it consist of 3 column without any column name. Like this:

	A	B	C
1	1	842079531126164	6.07572E+13
2	5	658583120618824	7.99964E+13
3	6	352578920202343	2.90854E+13
4	2	904016525281364	6.12204E+13
5	3	231979157207444	9.39894E+13
6	1	247922679700384	9.32678E+13
7	1	976198861625124	4.34897E+13
8	2	234541353291735	5.54717E+12

So as a solution I'm using x0, x1, and x2 as a column names such as:

	A	B	C
1	x0	x1	x2
2	1	8.42E+13	6.08E+13

2. Secondly second column consist of some tab spaces at the right end of the digits.



A	B	C	D
1	842079531126164	6.07572E+13	

So I had remove all these extra spaces from the data. After removing all these thing form the data. File will become like this

x0	x1	x2
1	8.42E+13	6.08E+13
5	6.59E+14	8.00E+13
6	3.53E+13	2.91E+13
2	9.04E+13	6.12E+13
3	2.32E+13	9.40E+13
1	2.48E+13	9.33E+13
1	9.76E+13	4.35E+13

Learning Outcomes:

From this exercise I had learn about the clustering how the selection of bad number of clusters effects the results and where we have to use elbow methods and where silhouette methods.

Explanation of Code:

As we have to perform clustering operation on the set of data that present inside the k-means.csv file so we have to read that data from the file by using panda library

```
import pandas as pd
fileData = pd.read_csv("k-means.csv")
fileData.head()
```

Next step is to analysis the data without doing anything so for this purpose we have to draw a scatter plot of the data.

```
#Now Trying to create a scatter graph of the file data
from matplotlib import pyplot as plot1
plot1.scatter(fileData.x0,fileData.x2)
plot1.xlabel('x0')
plot1.ylabel('x1')
plot1.title("Graph Between x0 and x1")
```

Second steps is to select the correct number of clusters so for this purpose firstly I'm selecting a random number of cluster for example firstly I had created 2 clusters as show in the code below

```
print("KMeans with 2 clusters by using x1 and x2")
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
predictedCluster_Y = kmeans.fit_predict(fileData[['x1','x2']])
# Now add the predicted cluster with the actual file data so
fileData['predictedCluster_Y'] = predictedCluster_Y;
fileData0 = fileData[fileData.predictedCluster_Y==0]
fileData1 = fileData[fileData.predictedCluster_Y==1]
# Now plost those graphs by using scatter graph function of matplotlib library
from matplotlib import pyplot as plot
plot.scatter(fileData0["x1"],fileData0['x2'],color='green')
plot.scatter(fileData1["x1"],fileData1['x2'],color='red')
plot.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],color='black',marker='X',label='Cluster Center');

plot.xlabel("x1");
plot.ylabel("x2");
plot.legend()
```

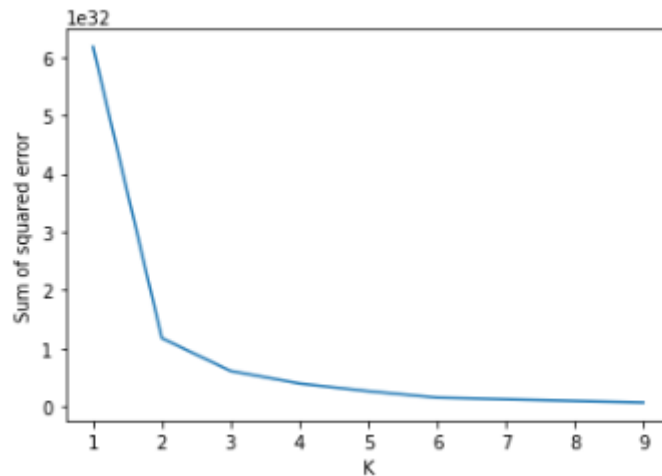
Third step is to repeat the second step with 4 clusters

Fourth steps is to use an elbow method for selecting a correct number of clusters

```
# Now you can see that the graph is ok so we have to use elbow method
sse = []
k_rng = range(1,10)
for k in k_rng:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(fileData[['x1','x2']])
    sse.append(kmeans.inertia_)
# Now creating a graph for the best fit cluster
plt.xlabel('K')
```

```
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

Image of the elbow graph:



Now we have to use 3 number of cluster for the data like this

```
# Now we have to apply kmeans with 3 cluster
kmeans=KMeans(n_clusters= 3, init = 'k-means++', max_iter = 100, n_init = 10, random_state = 0)
predictedCluster_Y = kmeans.fit_predict(fileData[['x1','x2']])
# Now add the predicted cluster with the actual file data so
fileData['predictedCluster_Y'] = predictedCluster_Y;
fileData0 = fileData[fileData.predictedCluster_Y==0]
fileData1 = fileData[fileData.predictedCluster_Y==1]
fileData2 = fileData[fileData.predictedCluster_Y==2]
# Now plost those graphs by using scatter graph function of matplotlib library
from matplotlib import pyplot as plot
plot.scatter(fileData0["x1"],fileData0["x2"],color='green')
plot.scatter(fileData1["x1"],fileData1["x2"],color='red')
plot.scatter(fileData2["x1"],fileData2["x2"],color='yellow')
plot.scatter(kmeans.cluster_centers_[0],kmeans.cluster_centers_[1],color='black',marker='X',label='Cluster
Center');
plot.xlabel("x1");
plot.ylabel("x2");
plot.legend()
```

So the next step is we have to calculate silhouette score the code of finding this is:

```
for n_clusters in range(0,10):# Maximum range should be 6, as it contains only 6 data points
    kmeans = KMeans(n_clusters=k,max_iter=100).fit(fileData[['x1','x2']])
    label = kmeans.labels_
    sil_coeff = silhouette_score(fileData[['x1','x2']],label,metric = 'euclidean')
    print('Cluster { }'.format(n_clusters))
    print('Coefficient { }'.format(sil_coeff))

# SO with the help of coefficient we can see the perfect number of cluster for the data is (3)
```

Achieved results:

As with the help of k-means clustering I had achieved that we can use elbow method instead of silhouette score secondly for the correct number of precision we must use 3 number of cluster for this data.