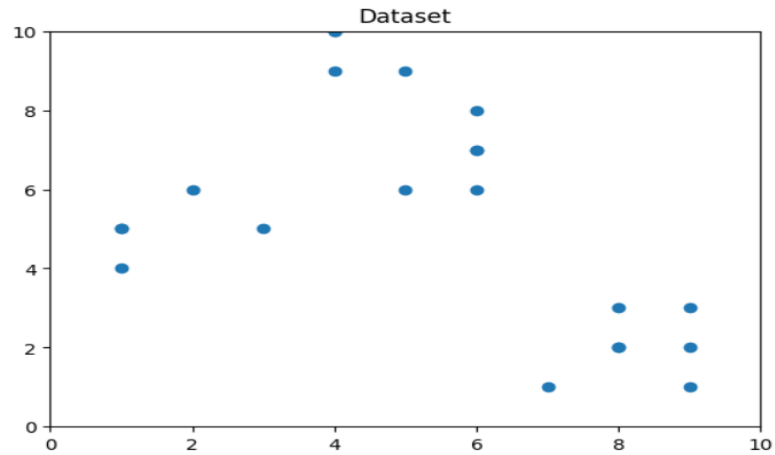
Problem Statement:

Implement the K-Means clustering algorithm to cluster a given dataset into K clusters. Choose an appropriate dataset and determine the optimal number of clusters using techniques such as the elbow method or silhouette score. Visualize the clustered data and analyze the quality of the clustering results

```
from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist
import numpy as np
import matplotlib.pyplot as plt
# Creating the data
x1 = np.array([3, 1, 1, 2, 1, 6, 6, 6, 5, 6,\
                                7, 8, 9, 8, 9, 9, 8, 4, 4, 5, 4])
x2 = np.array([5, 4, 5, 6, 5, 8, 6, 7, 6, 7, \
                                1, 2, 1, 2, 3, 2, 3, 9, 10, 9, 10])
X = np.array(list(zip(x1, x2))).reshape(len(x1), 2)
# Visualizing the data
plt.plot()
plt.xlim([0, 10])
plt.ylim([0, 10])
plt.title('Dataset')
plt.scatter(x1, x2)
plt.show()
```



---> **From the above visualization, we can see that the optimal number of clusters should be around 3.**

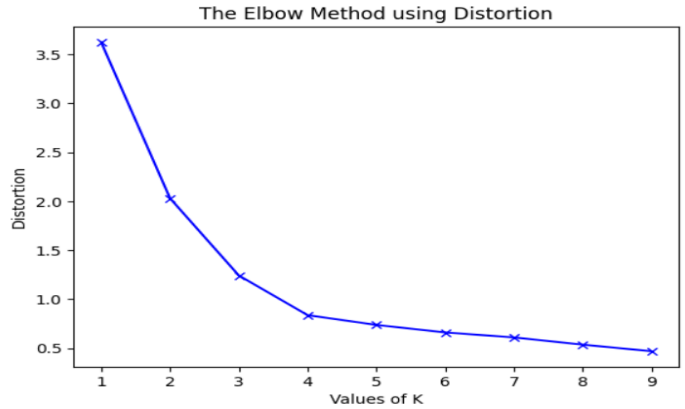
Distortion: It is calculated as the average of the squared distances from the cluster centers of the respective clusters to each data point. Typically, the Euclidean distance metric is used.

Distortion = 1/n * Σ(distance(point, centroid)^2)

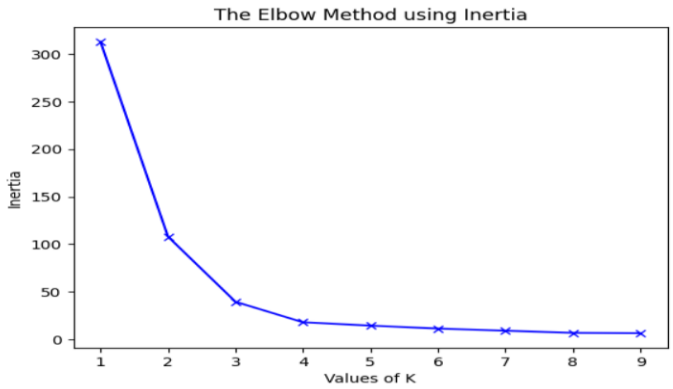
Inertia: It is the sum of the squared distances of samples to their closest cluster center.

Inertia = Σ(distance(point, centroid)^2

```
distortions = []

inertias = []

mapping1 = {}

mapping2 = {}

K = range(1, 10)

for k in K:

    # Building and fitting the model

    kmeanModel = KMeans(n_clusters=k).fit(X)

    kmeanModel.fit(X)

    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclidean'), axis=1))
/ X.shape[0])

    inertias.append(kmeanModel.inertia_)

    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclidean'), axis=1)) /
X.shape[0]

    mapping2[k] = kmeanModel.inertia_


for key, val in mapping1.items():

    print(f'{key} : {val}')

plt.plot(K, distortions, 'bx-')

plt.xlabel('Values of K')

plt.ylabel('Distortion')

plt.title('The Elbow Method using Distortion')

plt.show()
```




```
for key, val in mapping2.items():
        print(f'{key} : {val}')

plt.plot(K, inertias, 'bx-')

plt.xlabel('Values of K')

plt.ylabel('Inertia')

plt.title('The Elbow Method using Inertia')

plt.show()
```



The elbow method is a graphical method for finding the optimal K value in a k-means clustering algorithm. The elbow graph shows the within-cluster-sum-of-square (WCSS) values on the y-axis corresponding to the different values of K (on the x-axis). The optimal K value is the point at which the graph forms an elbow.

The distortion measures how spread out the clusters are, and the inertia measures how compact the clusters are.

The Elbow Method aims to find a value of k where the distortion or inertia starts to decrease at a slower rate, forming an elbow in the plot. The elbow point is around k=3 for both distortion and inertia, suggesting that 3 clusters may be optimal for this dataset.

-->Final Output code:

```
import matplotlib.pyplot as plt

# Create a range of values for k
```

```
k_range = range(1, 10)
# Initialize an empty list to
# store the inertia values for each k
inertia_values = []
# Fit and plot the data for each k value
for k in k_range:
        kmeans = KMeans(n_clusters=k, \ init='k-means++', random_state=42)
        y_kmeans = kmeans.fit_predict(X)
        inertia_values.append(kmeans.inertia_)
        plt.scatter(X[:, 0], X[:, 1], c=y_kmeans)
        plt.scatter(kmeans.cluster_centers_[:, 0],\ kmeans.cluster_centers_[:, 1], \ s=100, c='red')
        plt.title('K-means clustering (k={})'.format(k))
        plt.xlabel('Feature 1')
        plt.ylabel('Feature 2')
        plt.show()
# Plot the inertia values for each k
plt.plot(k_range, inertia_values, 'bo-')
plt.title('Elbow Method')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.show()
```