National University of Computer and Emerging Sciences

Lab Manual 03
Fundamentals of Big Data Lab

Course Instructor	Dr. Iqra Safdar
Lab Instructor (s)	Muhammad Mazarib M Aiss Shahid
Section	B1,B2
Semester	Spring 2023

K-means

K-means is an unsupervised learning method for clustering data points. The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster.

Here, we will show you how to estimate the best value for K using the elbow method, then use K-means clustering to group the data points into clusters.

How does it work?

First, each data point is randomly assigned to one of the K clusters. Then, we compute the centroid (functionally the center) of each cluster, and reassign each data point to the cluster with the closest centroid. We repeat this process until the cluster assignments for each data point are no longer changing.

K-means clustering requires us to select K, the number of clusters we want to group the data into. The elbow method lets us graph the inertia (a distance-based metric) and visualize the point at which it starts decreasing linearly. This point is referred to as the "eblow" and is a good estimate for the best value for K based on our data.

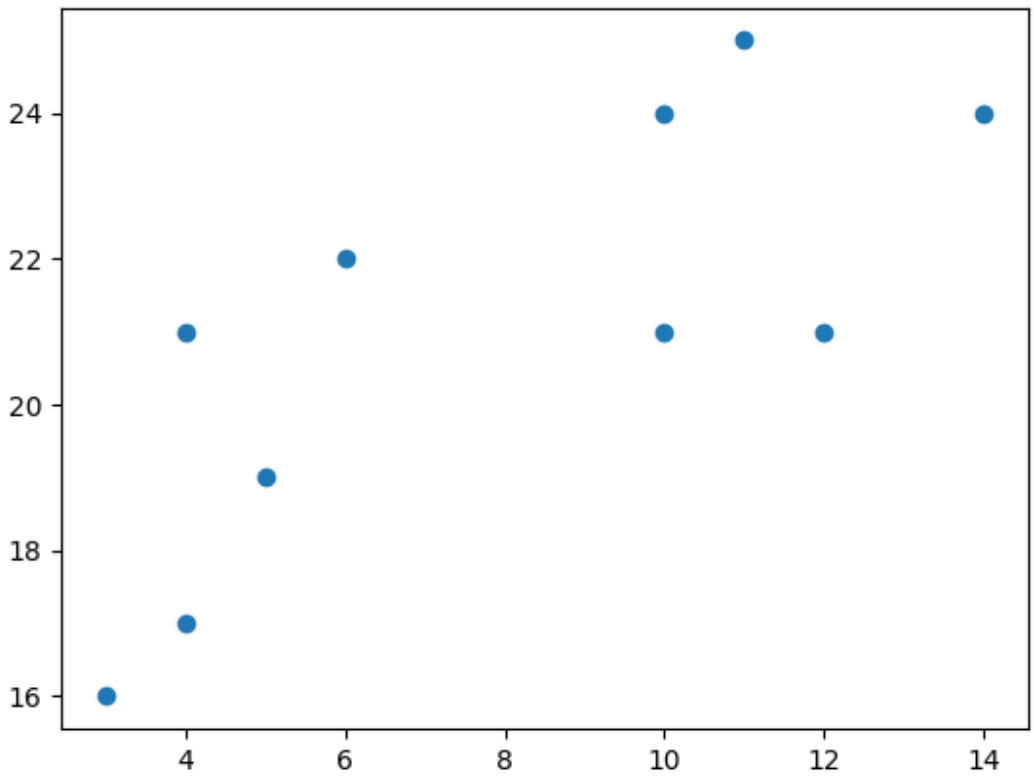
Example

Start by visualizing some data points:

```
import matplotlib.pyplot as plt



x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]



plt.scatter(x, y)
```

```
plt.show()
```

Result

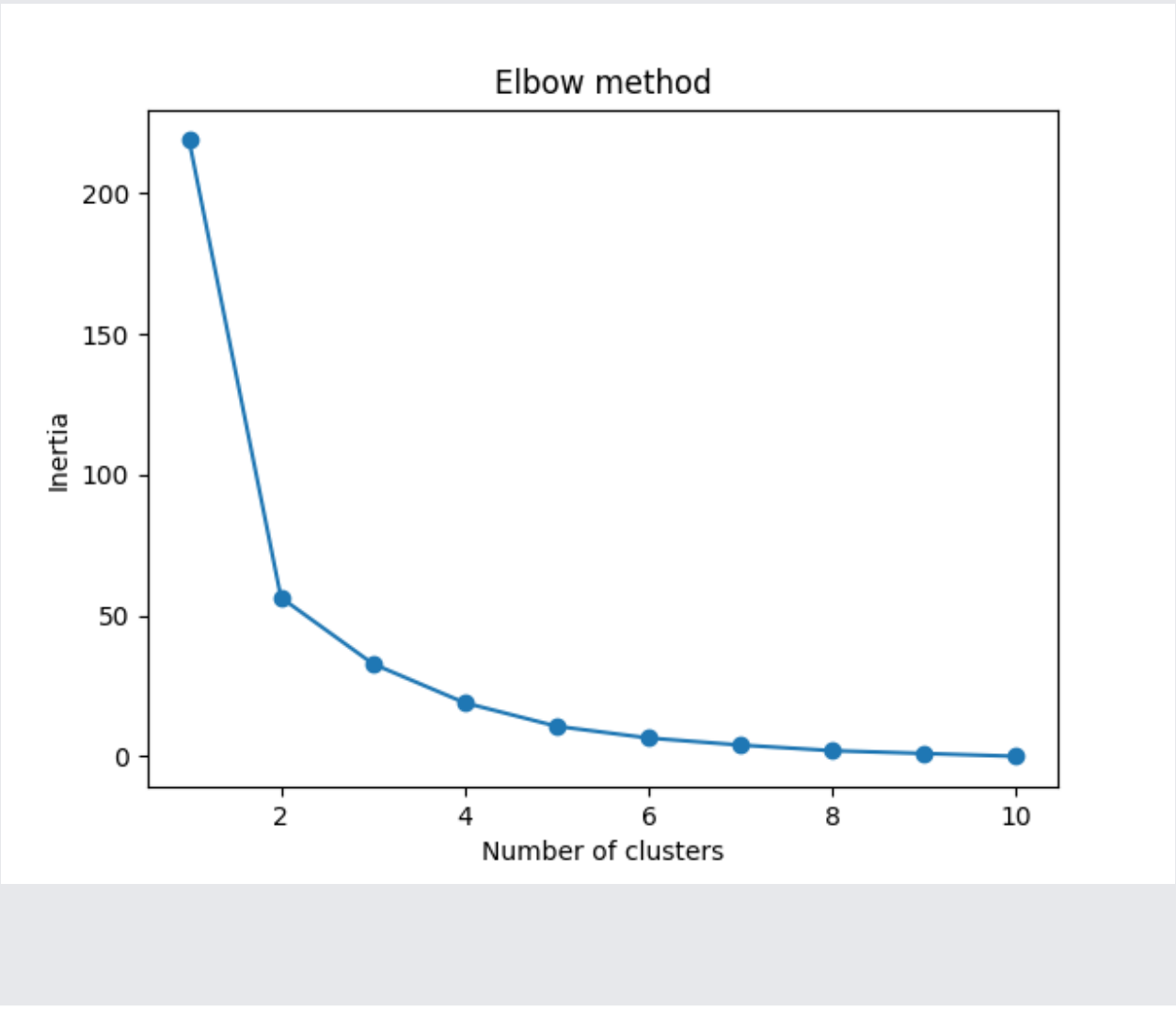


Now we utilize the elbow method to visualize the intertia for different values of K:

Example

```
from sklearn.cluster import KMeans



data = list(zip(x, y))

inertias = []



for i in range(1,11):

    kmeans = KMeans(n_clusters=i)

    kmeans.fit(data)

    inertias.append(kmeans.inertia_)



plt.plot(range(1,11), inertias, marker='o')

plt.title('Elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()
```

Result



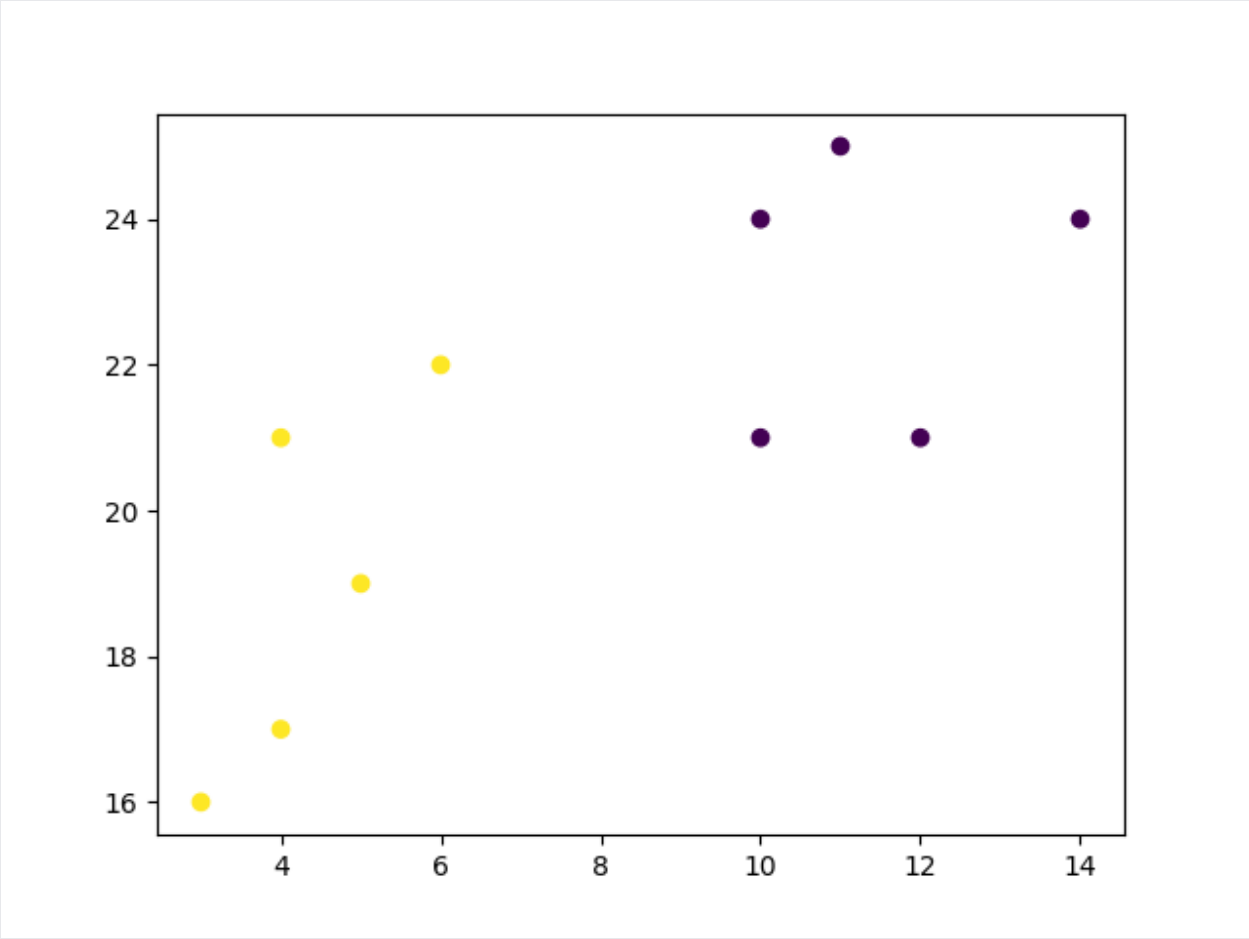
The elbow method shows that 2 is a good value for K, so we retrain and visualize the result:

Example

```
kmeans = KMeans(n_clusters=2)

kmeans.fit(data)



plt.scatter(x, y, c=kmeans.labels_)
```

```
plt.show()
```

Result



Example Explained

Import the modules you need.

```
import matplotlib.pyplot as plt


from sklearn.cluster import KMeans
```

scikit-learn is a popular library for machine learning.

Create arrays that resemble two variables in a dataset. Note that while we only use two variables here, this method will work with any number of variables:

```
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]


y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
```

Turn the data into a set of points:

```
data = list(zip(x, y))


print(data)
```

Result:

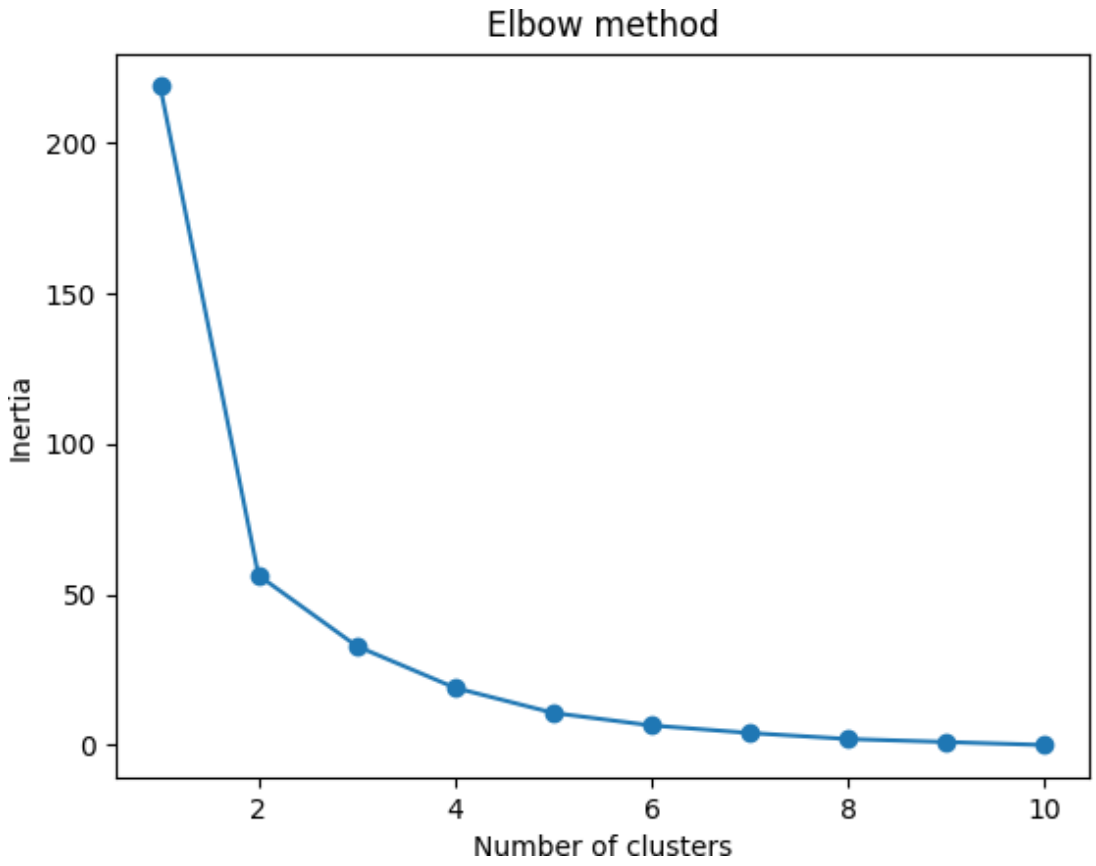
```
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6,
22), (10, 21), (12, 21)]
```

In order to find the best value for K, we need to run K-means across our data for a range of possible values. We only have 10 data points, so the maximum number of clusters is 10. So for each value K in range(1,11), we train a K-means model and plot the intertia at that number of clusters:

```
inertias = []




for i in range(1,11):


    kmeans = KMeans(n_clusters=i)
```

```
    kmeans.fit(data)

    inertias.append(kmeans.inertia_)



plt.plot(range(1,11), inertias, marker='o')

plt.title('Elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()
```

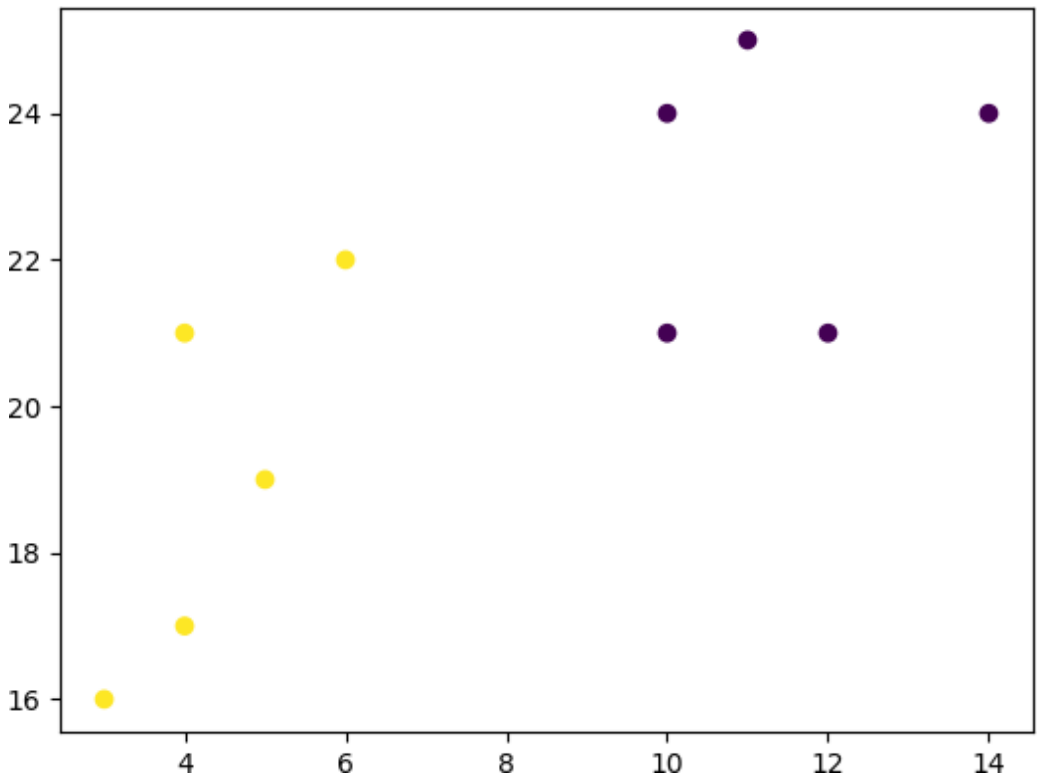
Result:



We can see that the "elbow" on the graph above (where the interia becomes more linear) is at K=2. We can then fit our K-means algorithm one more time and plot the different clusters assigned to the data:

```
kmeans = KMeans(n_clusters=2)

kmeans.fit(data)



plt.scatter(x, y, c=kmeans.labels_)

plt.show()
```

Result:



Task A:

1. Use built in k-means function to work on given data, use annual income and spending score.

Task B:

1. Design your own K-means clustering algorithm, use annual income and spending score.