Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exe25 - k-Means Exercise

Name:

IC Number:

Date :

Introduction :

Conclusion :

Exercise 1: Build and Plot k-Means

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Step 1: create blobs with the size of 500, and center of 3

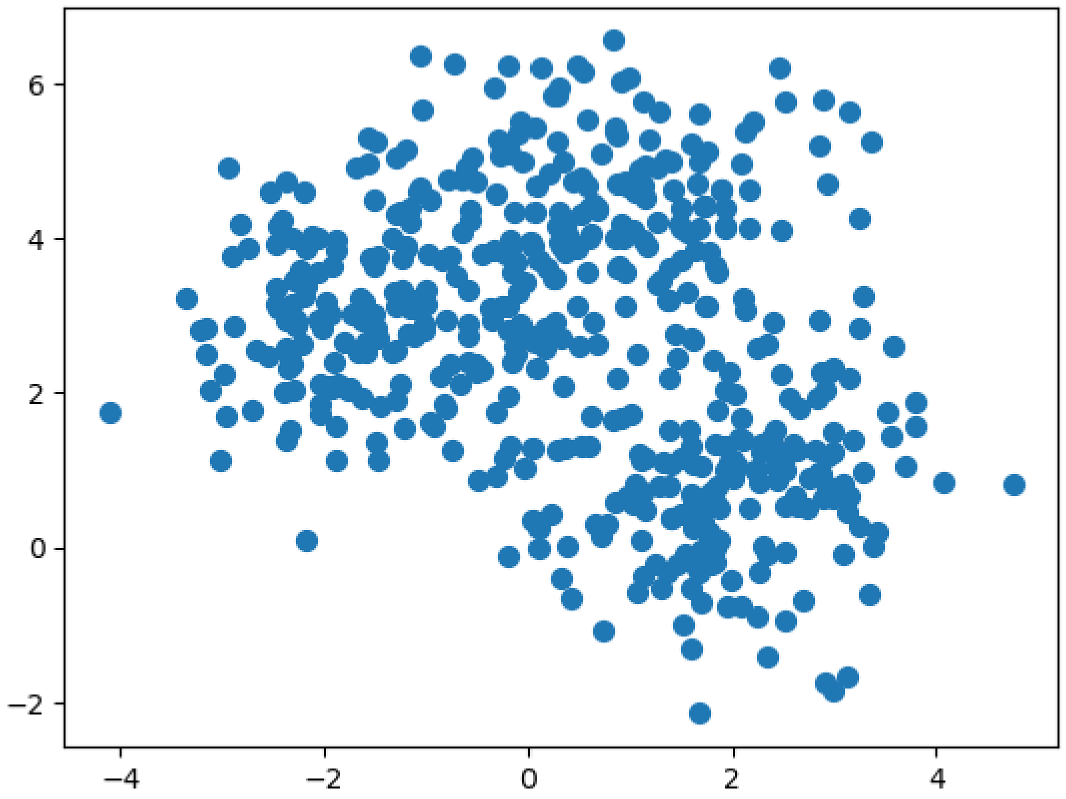
In [4]:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=500, centers=3, n_features=2,random_state=0)
```

Step 2: Plot the distribution of the blobs

In [5]:

```
plt.scatter(X[:, 0], X[:, 1], s=50);
```



Step 3: Use K-means, find the centers of these clusters

In [6]:

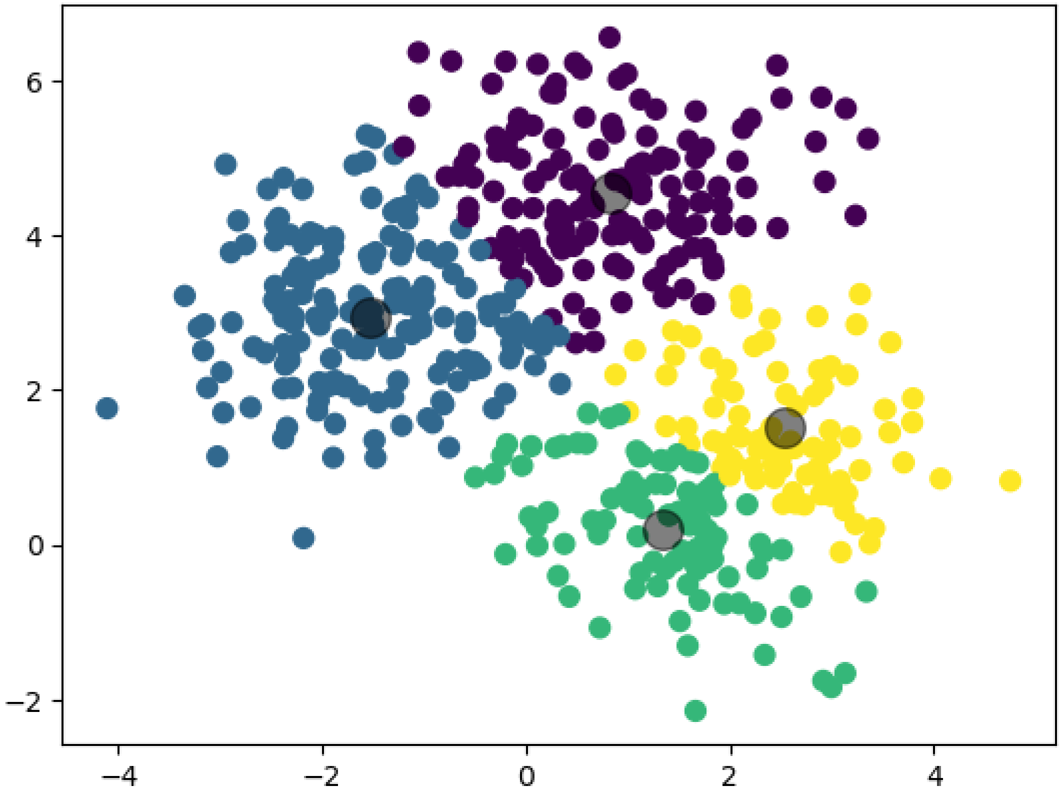
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

```
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1412: FutureWarning: The default value of `n_init` will change from 1
0 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the w
arning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

Step 4: Plot the blobs with the found centers

In [7]:

```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```



Additional/Optional:

Step 5: How can you find out the automatically assigned "labels" in the produced clusters?

In [8]:

```
kmeans.labels_
```

Out[8]:

```
array([0, 1, 1, 3, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 2, 0, 1, 0, 3, 2,
       3, 1, 0, 1, 2, 1, 0, 1, 1, 0, 0, 1, 2, 2, 3, 3, 2, 2, 1, 0, 0, 3,
       3, 0, 2, 0, 1, 1, 0, 0, 0, 0, 1, 1, 2, 2, 0, 2, 3, 1, 2, 2, 3, 1,
       3, 2, 1, 0, 1, 0, 0, 1, 3, 1, 3, 2, 2, 0, 0, 0, 1, 0, 1, 2, 1, 1,
       1, 1, 0, 0, 3, 0, 1, 3, 1, 0, 1, 0, 3, 1, 1, 0, 0, 2, 0, 3, 0, 2,
       0, 0, 2, 1, 1, 2, 1, 3, 2, 1, 1, 3, 0, 1, 1, 0, 1, 0, 1, 0, 3, 0,
       1, 2, 0, 3, 1, 1, 3, 0, 0, 0, 1, 2, 3, 0, 0, 2, 3, 0, 3, 0, 0, 0,
       3, 0, 1, 2, 1, 0, 3, 3, 1, 1, 1, 3, 0, 3, 0, 3, 1, 1, 3, 0, 0, 1,
       1, 2, 3, 2, 3, 2, 1, 1, 2, 3, 0, 3, 1, 1, 1, 0, 3, 3, 0, 1, 2, 0,
       1, 2, 1, 3, 0, 0, 1, 2, 2, 2, 1, 1, 0, 2, 2, 1, 1, 0, 1, 0, 1, 1,
       3, 1, 3, 2, 0, 0, 0, 3, 1, 0, 0, 1, 1, 2, 1, 1, 0, 2, 0, 0, 0, 3,
       2, 2, 0, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0, 2, 1, 2, 3, 0, 0, 3,
       2, 0, 1, 3, 1, 3, 3, 1, 3, 0, 1, 3, 0, 3, 2, 2, 2, 1, 3, 0, 3, 0,
       0, 1, 1, 2, 3, 1, 1, 0, 0, 3, 1, 0, 1, 3, 0, 3, 1, 2, 0, 1, 0, 2,
       0, 1, 2, 1, 1, 0, 1, 0, 3, 1, 1, 0, 0, 2, 0, 2, 0, 0, 0, 3, 0, 1,
       3, 3, 2, 1, 2, 2, 0, 1, 1, 1, 2, 1, 3, 3, 0, 0, 3, 0, 1, 1, 2, 0,
       1, 1, 1, 3, 3, 2, 2, 1, 0, 2, 1, 0, 0, 0, 2, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 2, 2, 1, 3, 0, 1, 0, 1, 0, 0, 1, 2, 1, 1, 2, 0, 3, 0, 3,
       1, 1, 0, 3, 3, 0, 2, 2, 1, 3, 3, 1, 3, 1, 0, 0, 1, 2, 2, 0, 0, 2,
       1, 1, 0, 1, 3, 0, 1, 0, 3, 0, 1, 0, 0, 2, 0, 0, 0, 0, 2, 0, 2, 1,
       2, 1, 0, 1, 3, 1, 3, 1, 2, 0, 1, 3, 2, 3, 2, 3, 0, 1, 3, 1, 0, 0,
       2, 1, 1, 1, 1, 2, 0, 0, 3, 2, 1, 3, 1, 2, 2, 2, 2, 0, 0, 2, 2, 0,
       2, 1, 1, 0, 2, 2, 1, 0, 1, 0, 3, 3, 3, 3, 0, 3])
```

Step 6: How about classes? How to find out where there are classes.

In [9]:

```
kmeans.n_clusters
```

Out[9]:

```
4
```

Exercise 2: k-Means with the Iris dataset

Step 1: Load the iris dataset from sklearn and other necessary libraries

In [31]:

```
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
```

Step 2: Set the training and target data as X and y respectively. Display the targets.

In [32]:

```
X = iris.data
y = iris.target
```

Introducing - *the Elbow Method:* A technique to allow you to identify the best K

General idea: iterate the creation of k-Means clusters with increasing sizes, and record down the value of kmeans.inertia_ (inertia_: Sum of squared distances of samples to their closest cluster center.)

Step 3: create a list named wcss and store the inertia values for a selected range of ks.

In [33]:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)

kmeans.fit(X)
wcss = kmeans.inertia_
wcss
```

```
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1412: FutureWarning: The default value of `n_init` will change from 1
0 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the w
arning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[33]:

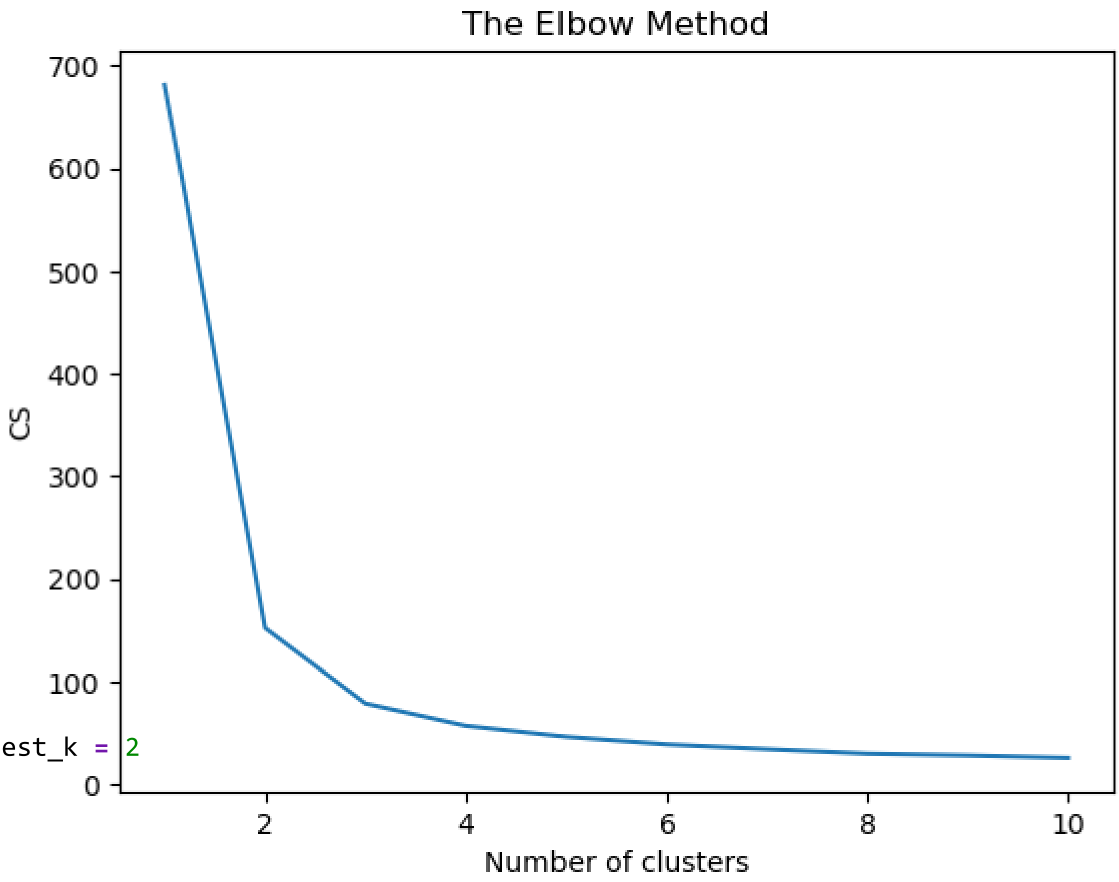
```
152.34795176035792
```

Step 4: Plot a graph to look at 'The elbow'

In [34]:

```
from sklearn.cluster import KMeans
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, ran
    kmeans.fit(X)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 11), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```

```
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

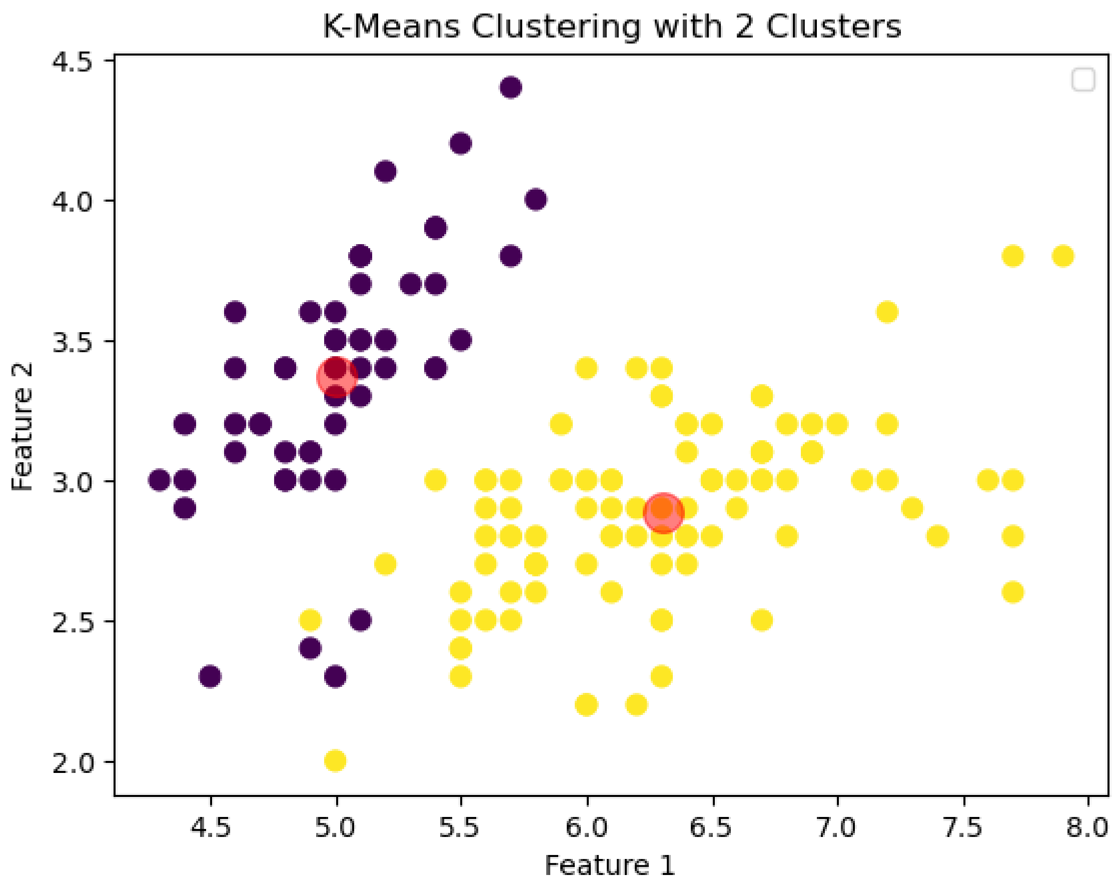

The Elbow Method
CS
700
600
500
400
300
200
100
0
2
4
6
8
10
Number of clusters

```
best_k = 2
```

In [41]:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Best K value obtained from the Elbow Method


# Initialize and fit the KMeans model with the best K value
kmeans = KMeans(n_clusters=best_k, init='k-means++', max_iter=300, n_init=10, random_sta
y_kmeans = kmeans.fit_predict(X)

# Visualize the clusters using a scatter plot
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5);
# Plot the centroids of the clusters
plt.title('K-Means Clustering with 2 Clusters')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```

```
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```



Additional/Optional:

Step 7: Plot the actual and Predicted side by side

In [ ]:

```

```