Computer Engineering Department National University of Technology Islamabad, Pakistan

Introduction to Data Mining

Practice Exercise 08

Name: Muhammad Sami Uddin Rafay

Roll Number: F18604013

Submitted To: Dr. Kamran Javed

Date: 31 January 2020

Practice Exercise 08

Unsupervised Machine Learning | K-mean Clustering

Objective:

- Implement K-means Clustering Algorithm in Python.

Equipment/Software Required:

- Python (Spyder 4.0 Anaconda Distribution)

Tasks:

1. Explain what is K-means clustering and how we choose the number of clusters?
2. Write main steps of K-means clustering
3. Reproduce the given code of K-means clustering (using clust.xlsx file) and explain/comment each line of code i.e., input, output attributes and function.
4. Plot 2D and 3D views of given data and label both figures.
5. Implement K-means for fisher iris data (Load fisher iris dataset with petal lengths and petal widths)
6. Determine the number of clusters for fisher data.
7. Plot clusters and cluster centers for each cluster

Answers:

1. **Explain what is K-means clustering and how we choose the number of clusters?**

   K-means clustering is a simple unsupervised machine learning algorithm which the modern data scientists mostly use for discovering relevant and hidden patterns in non-labeled data. All its need a way to compare observations, a way to guess how many clusters exists in the data, and a way to calculate averages for each cluster it predicts.

   To determine the optimal number of clusters, we have to select the value of k at the "elbow". Plot of sum of squared distances for k in the range specified above. If the plot looks like an arm, then the elbow on the arm is optimal k.

2. **Write main steps of K-means clustering.**

   Main Steps of K-means clustering:

   i. Initialize the cluster centers
   ii. Assign observations to the closest cluster center
   iii. Once we have these cluster centers, we can assign each point to the cluster based on minimum distance to cluster center.
   iv. Revise cluster centers as mean o assigned observations.

   Basic K-means Algorithm:

   i. Select K points as initial centroids
   ii. Repeat
   iii. From k clusters by assigning each point to its closest centroid
   iv. Recompute the centroids of each cluster.
   v. Until centroids do not change

The rest of answers are implemented below: -

Code (clust.xlsx):

 # importing libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
```

 # Loading clust.xlsx dataset

```
clust=pd.read_excel(r"C:\Users\User\Desktop\clust.xlsx")
```

 # Specifying Columns name

```
clust=pd.DataFrame(clust)
columns=["X", "Y", "Z"]
clust.columns=columns
#print(clust)
X=clust["X"]
Y=clust["Y"]
Z=clust["Z"]
X1=np.array(list(zip(X,Y,Z)))
```

 # 2D plotting of Dataset

```
plt.figure(1)
plt.scatter(X,Y)
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(True)
```

 # 3D plotting of Dataset

```
fig2 = plt.figure(2)
plot2 = Axes3D(fig2)
plot2.scatter(X, Y, Z)
plot2.set_xlabel("X")
plot2.set_ylabel("Y")
plot2.set_zlabel("Z")
```

 # Determining no. of clusters using Elbow Method

```
plt.figure(3)
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12))
visualizer.fit(clust)        # Fit the data to the visualizer
visualizer.show()
```

 # Using K-means command of sklearn.cluster

```
# Number of clusters
kmeans = KMeans(n_clusters=4)
# Fitting the input data
kmeans = kmeans.fit(X1[:,:3])
# Getting the cluster labels
labels = kmeans.predict(X1[:,:3])
# Centroid values
centroids = kmeans.cluster_centers_
kmeans.labels_
print(centroids)
#print(labels)
print(kmeans)
print(kmeans.labels_)

# 3D Plotting of clusters and cluster centers for each cluster

fig3 = plt.figure(4)
plot3 = Axes3D(fig3)
plot3.scatter(X1[:, 0], X1[:, 1], X1[:,2],c=kmeans.labels_,cmap='rainbow')
plot3.scatter(centroids[:, 0], centroids[:, 1], centroids[:, 2], marker='*', c='k', s=500)
plot3.set_xlabel("X")
plot3.set_ylabel("Y")
plot3.set_zlabel("Z")
plt.show()
```

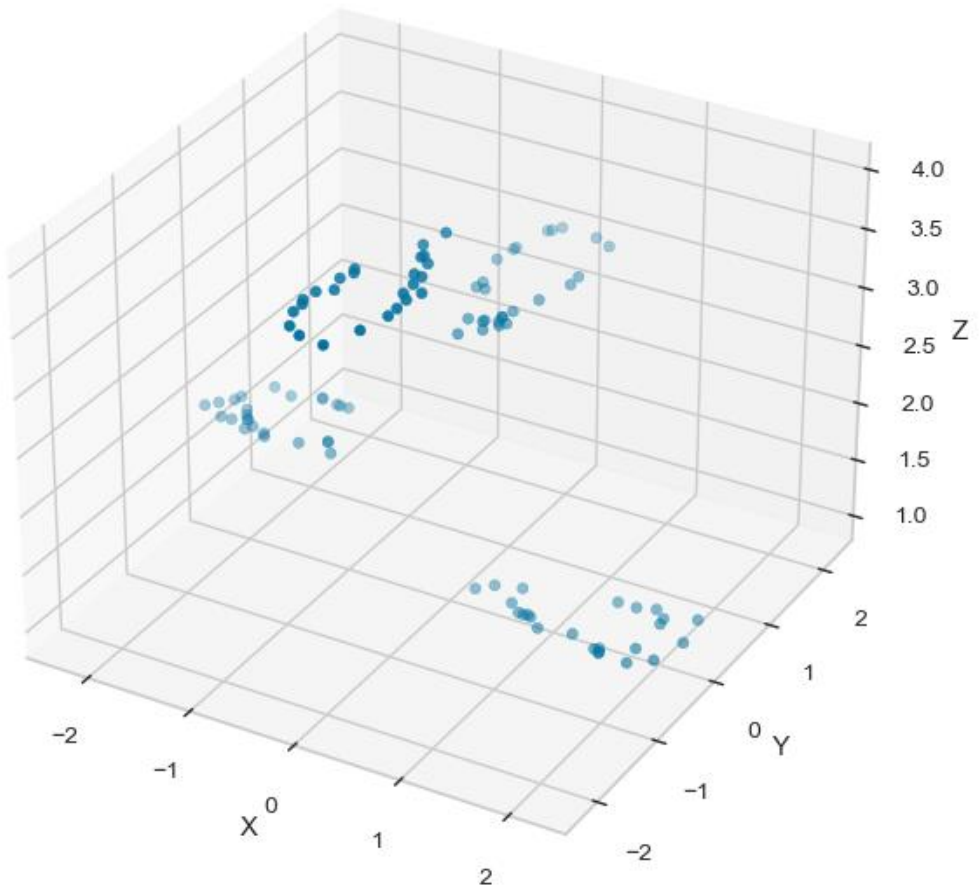
Output:

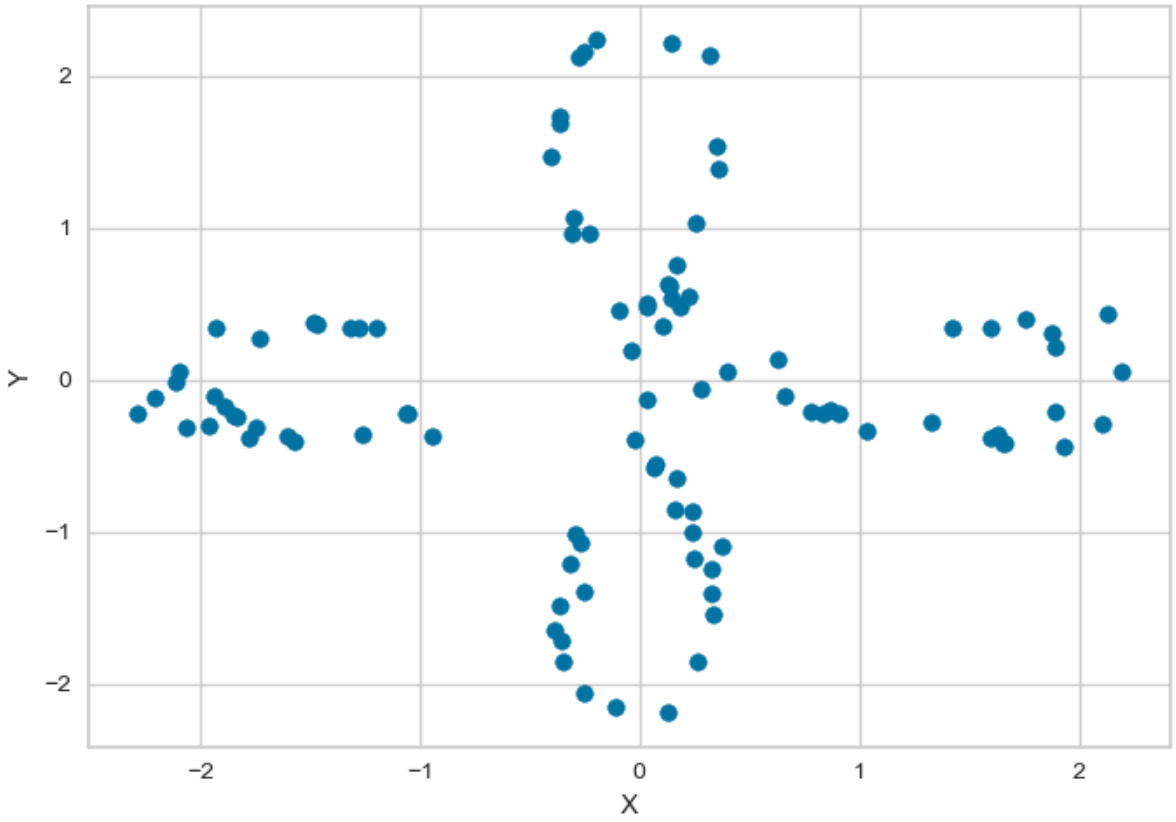
Centeroids : [[1.19098516e-03 -1.24666367e+00 4.00000000e+00]
 [-1.66510021e+00 -7.77102678e-02 2.00000000e+00]
 [1.37801031e+00 -7.84103778e-02 1.00000000e+00]
 [-8.93909608e-03 1.13073199e+00 3.00000000e+00]]

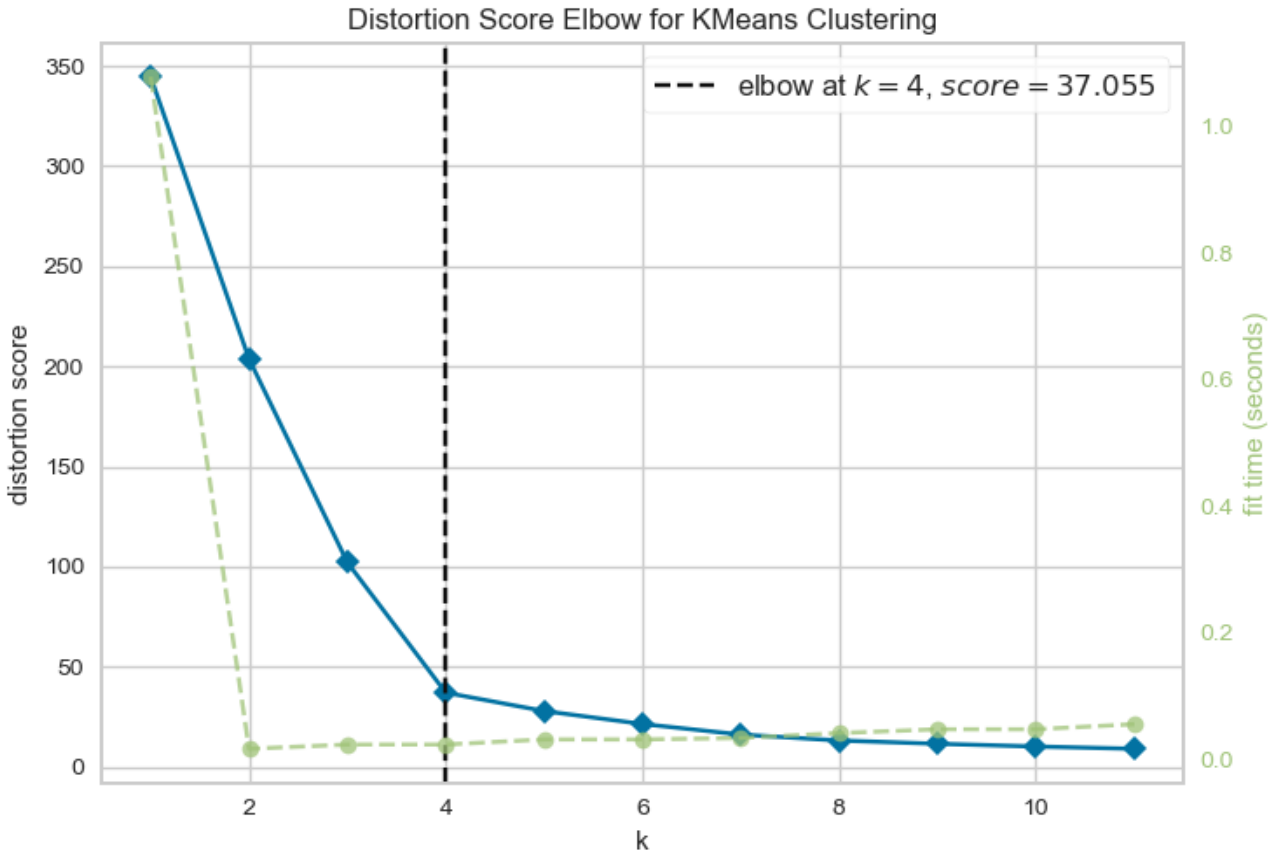
KMeans(n_clusters=4)

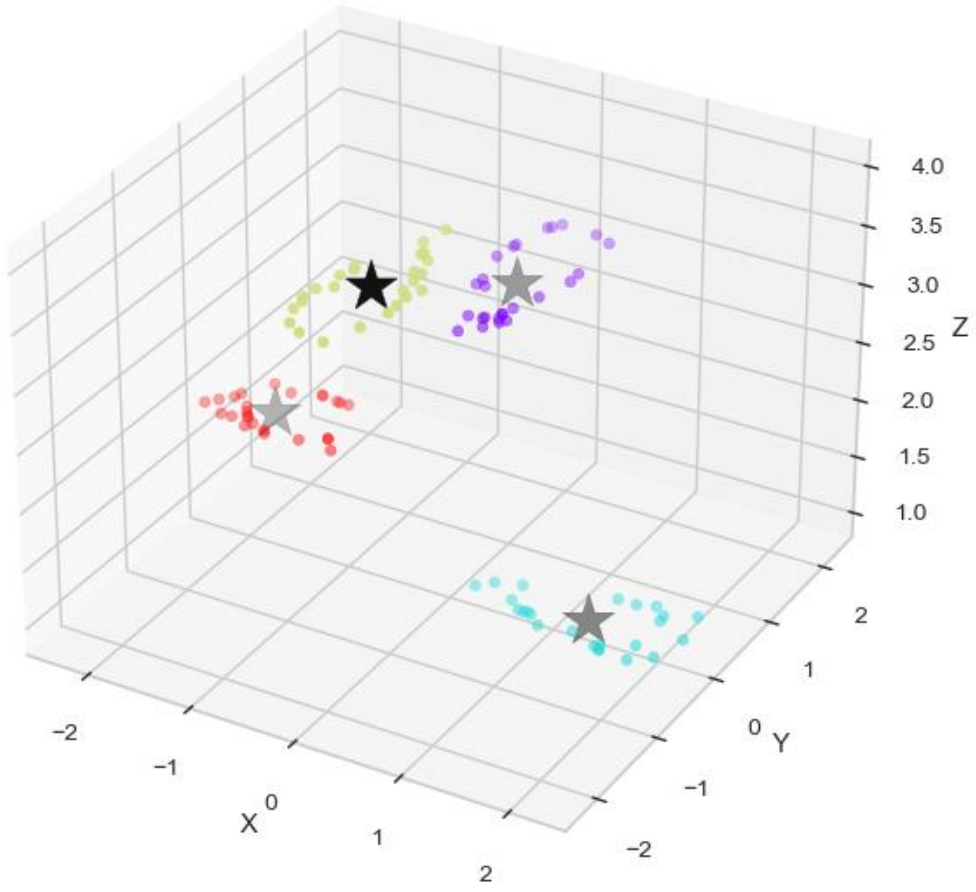
Labels: [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Graphs:






Distortion Score Elbow for KMeans Clustering
elbow at $k = 4$, $score = 37.055$
distortion score
fit time (seconds)
k
350
300
250
200
150
100
50
0
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10


X
Y
Z
4.0
3.5
3.0
2.5
2.0
1.5
1.0
−2
−1
0
1
2

Code (fisher iris):

 # importing libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
```

 # Loading dataset

```
fisher_iris=pd.read_csv(r"C:\Users\User\Downloads\iris.data")
```

 # Specifying Columns name

```
attributes = ["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]
fisher_iris.columns = attributes
print("Head of Fisher Dataset : \n",fisher_iris.head())
petal_length=fisher_iris["petal_length"]
petal_width=fisher_iris["petal_width"]
sepal_length=fisher_iris["sepal_length"]
sepal_width=fisher_iris["sepal_width"]
X=np.array(list(zip(petal_length,petal_width, sepal_width,sepal_length)))
```

 # 2D plotting of Petal Length and Petal Width

```
plt.figure(1)
plt.scatter(petal_length, petal_width)
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.grid(True)
plt.show()
```

 # Determining no. of clusters Elbow Method

```
plt.figure(2)
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12))
visualizer.fit(X)        # Fit the data to the visualizer
visualizer.show()
```

 # 3D plotting of Petal Length, Petal Width and Sepal width

```
fig = plt.figure(3)
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:,2])
ax.set_xlabel("petal Length")
ax.set_ylabel("petal width")
ax.set_zlabel("sepal width")
```

 # Using K-means command of sklear.cluster

 # Number of clusters

```
kmeans = KMeans(n_clusters=3)
# Fitting the input data
kmeans = kmeans.fit(X[:,:4])
# Getting the cluster labels
labels = kmeans.predict(X[:,:4])
# Centroid values
centroids = kmeans.cluster_centers_
kmeans.labels_
print("Centroids : ",centroids)
print("Cluster Labels : ",labels)
print(kmeans)
#print(kmeans.labels_)

# 3D Plotting of clusters and cluster centres for each cluster

fig2 = plt.figure(4)
bx = Axes3D(fig2)
bx.scatter(X[:, 0], X[:, 1], X[:,2],c=kmeans.labels_,cmap='rainbow')
bx.scatter(centroids[:, 0], centroids[:, 1], centroids[:, 2], marker='*', c='k', s=500)
bx.set_xlabel("petal Length")
bx.set_ylabel("petal width")
bx.set_zlabel("sepal width")
plt.show()
```

Output:

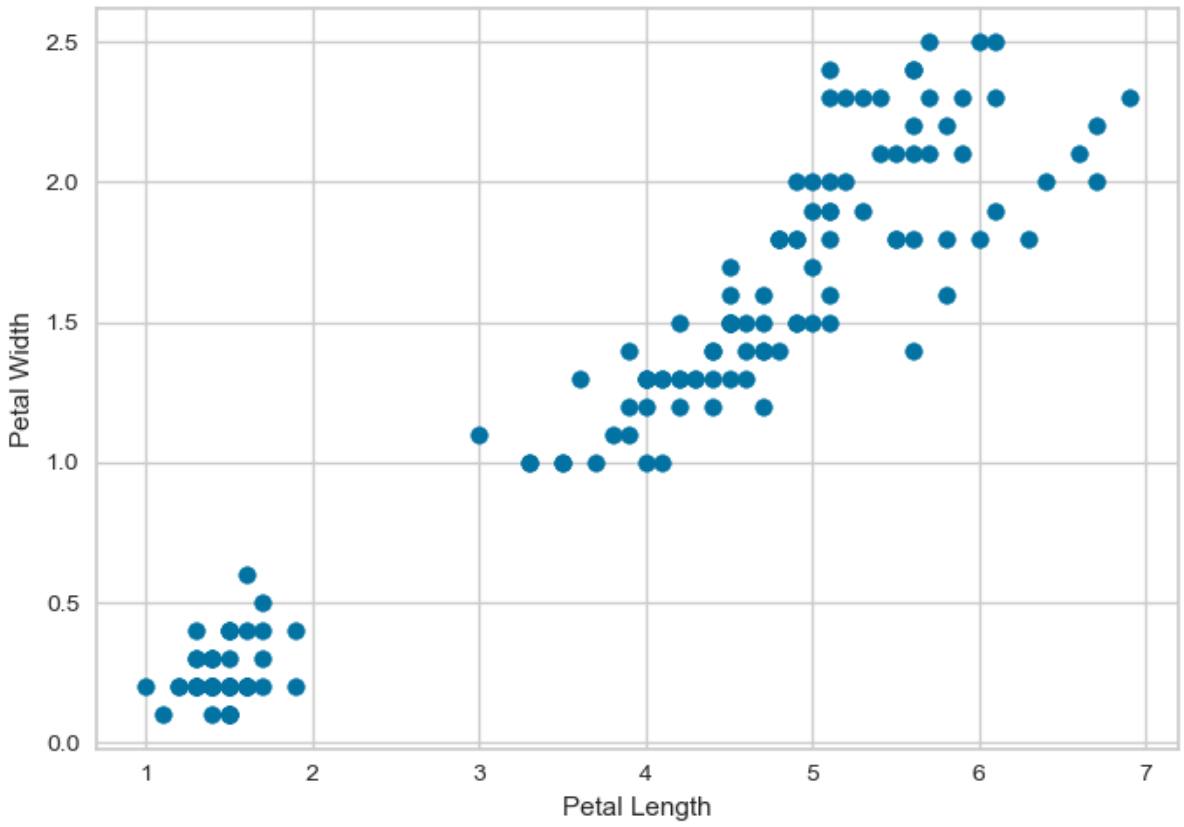
Head of Fisher Dataset :

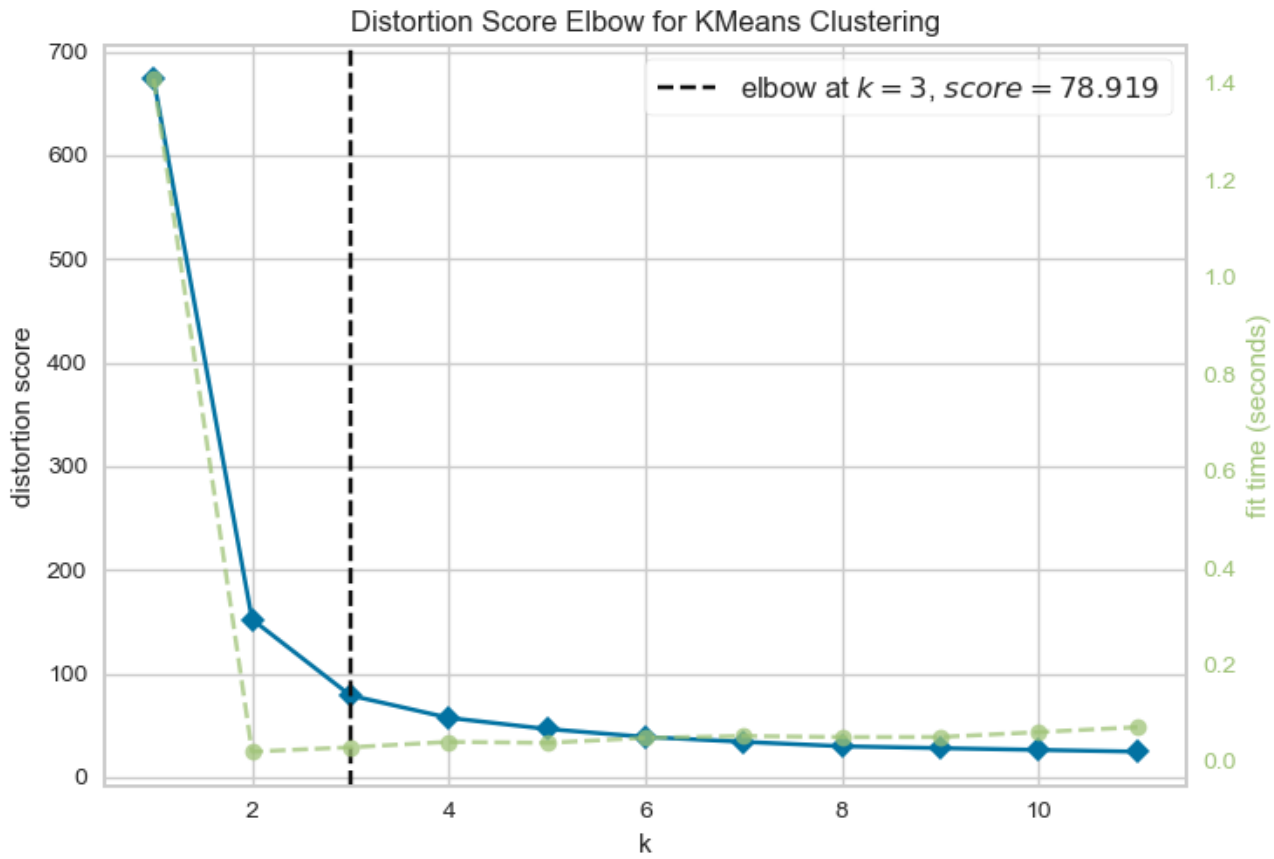
	sepal_length	sepal_width	petal_length	petal_width	class
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

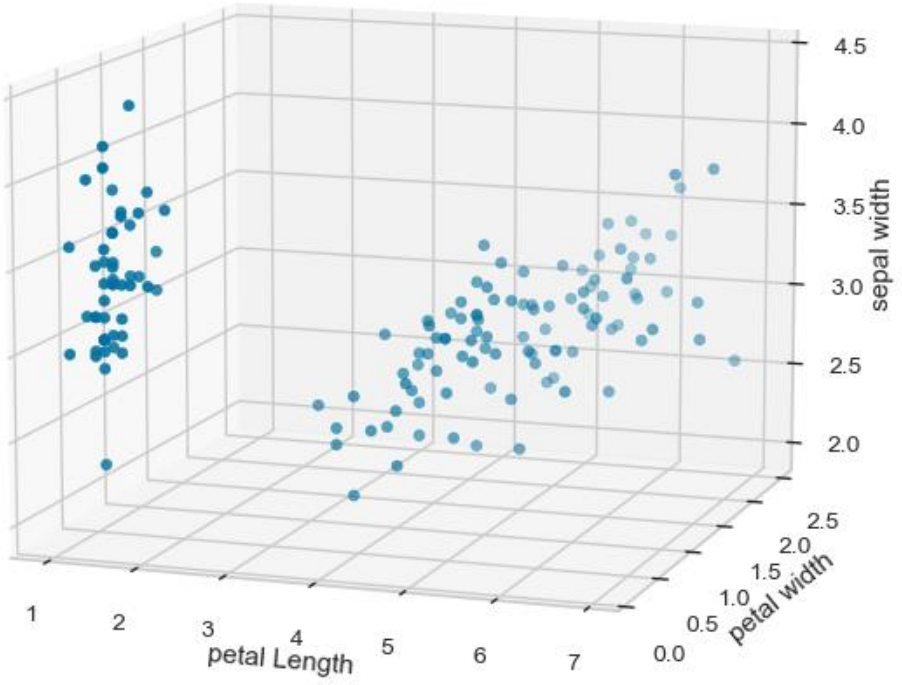
Centroids :
```
[[1.46530612 0.24489796 3.41632653 5.00408163]
 [4.39354839 1.43387097 2.7483871  5.9016129 ]
 [5.74210526 2.07105263 3.07368421 6.85      ]]
```
Cluster Labels :
```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2 2
 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2 2
 1]
```

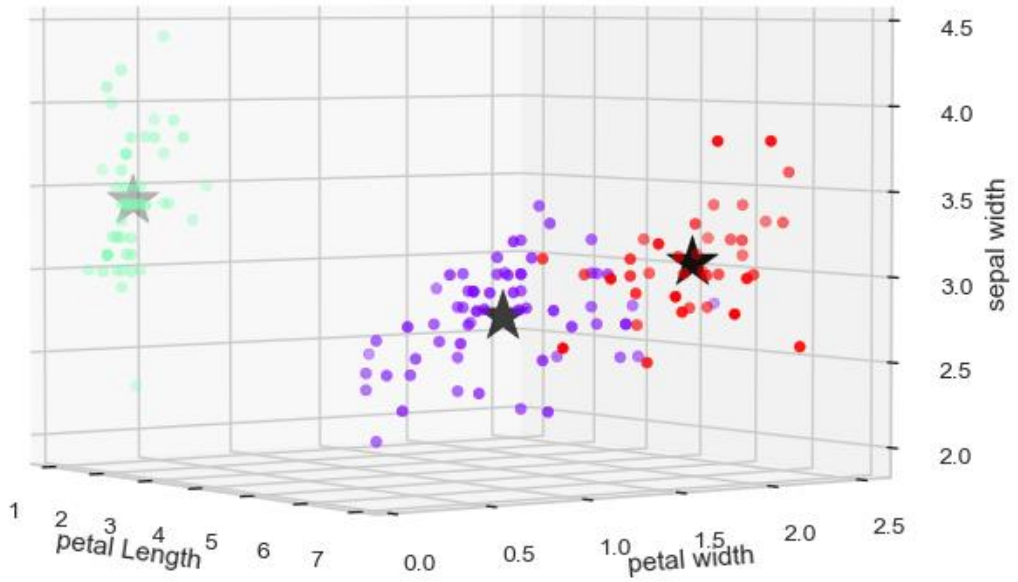
KMeans(n_clusters=3)

Graphs:


Petal Width
Petal Length
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


Distortion Score Elbow for KMeans Clustering
elbow at $k = 3$, $score = 78.919$
distortion score
fit time (seconds)
k
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
2
4
6
8
10


4.5
4.0
3.5
3.0
2.5
2.0
sepal width
2.5
2.0
1.5
1.0
0.5
0.0
petal width
1
2
3
4
5
6
7
petal Length


4.5
4.0
3.5
3.0
2.5
2.0
sepal width
1
2
3
4
5
6
7
petal Length
0.0
0.5
1.0
1.5
2.0
2.5
petal width

Results and Discussions:

As we discussed in class that k-means is useful to find hidden patterns in data and here we proved it by applying this technique to two different dataset and calculated beneficial results for further analysis. One Dataset appeared with four clusters and the other one appeared with 3 clusters with reasonable outcomes.

Conclusion:

The k-means clustering technique is simple quick algorithm that can be applied to large datasets to separate them into different partitions.