Exercise 8: Implement K Means clustering algorithm

Document dataset:

(a) IRIS dataset D4: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/iris.scale

(b) rcv1v2 (topics; subsets D5:https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html#rcv1v2(topics subsets)

1: Implement K Means clustering algorithm (using any software library). You should use D4 or D5 datasets. Also, you should also choose a criterion for selecting an optimal value of k (number of clusters).

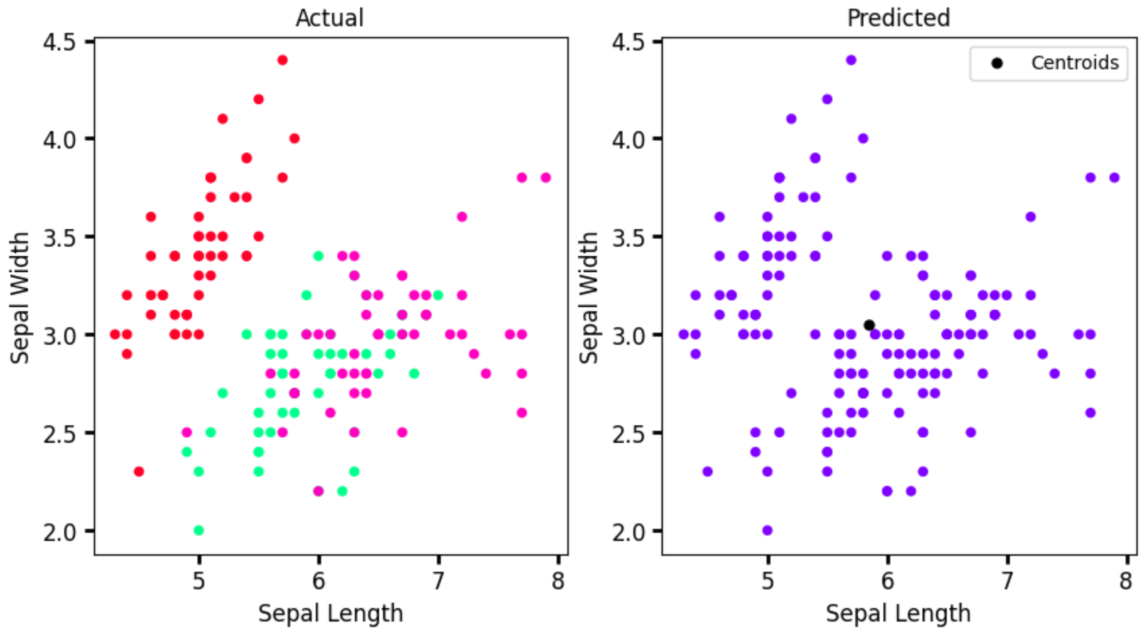
2: Cluster the data set D4 or D5 using your own implementation of K-means algorithm. Show the results.

In [ ]:

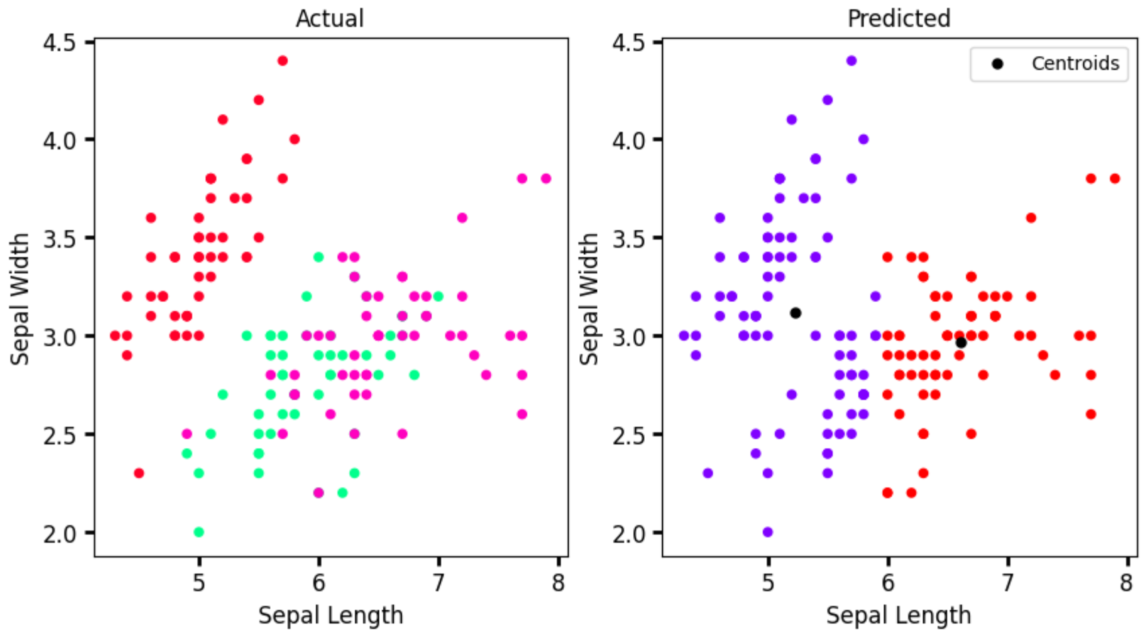
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random
import warnings
warnings.filterwarnings('ignore')
import matplotlib.cm as cm

# Input: Dataset
df = pd.read_csv('iris.data')

iris_datasets = np.array(df)

# Taking only two attributes of the Dataset
test_x = iris_datasets[:,[0,1]]

# Storing the Actual Target value in a new Array
actuallabel = iris_datasets[:,-1:]

for iters in range(0,149):
    if actuallabel[iters] == 'Iris-setosa':
        actuallabel[iters] = 0
    elif actuallabel[iters] == 'Iris-versicolor':
        actuallabel[iters] = 1
    elif actuallabel[iters] == 'Iris-virginica':
        actuallabel[iters] = 2

Actual_Value = []
for iters in range(len(actuallabel)):
    Actual_Value.append(actuallabel[iters][0])

# K Means Clustering Algorithm
sse = []
for k in range(1, 11):                # Running the Model for different v
```

```
    # Create K Means Clustering object and Train it over the Dataset
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(test_x)

    # Centers obtained after training of model
    centers = kmeans.cluster_centers_

    # Output: The Cluster Centers obtained
    print("\n\nFor k = ",k,"\n")
    for k in range(k):
        print("  Center ",k+1," = ",centers[k])

    # Calculating the sum of distances of samples to their closest cluste
    sse.append(kmeans.inertia_)

    # Output: The Actual Cluster Plot vs Predicted Cluster Plot for given
    fig, axes = plt.subplots(1, 2, figsize=(10,5))
    axes[0].scatter(test_x[:, 0], test_x[:, 1], c=actuallabel,
                    cmap='gist_rainbow', s=20)
    axes[1].scatter(test_x[:, 0], test_x[:, 1], c=kmeans.labels_,
                    cmap='rainbow', s=20)
    axes[0].set_xlabel('Sepal Length', fontsize=12)
    axes[0].set_ylabel('Sepal Width', fontsize=12)
    axes[1].set_xlabel('Sepal Length', fontsize=12)
    axes[1].set_ylabel('Sepal Width', fontsize=12)

    axes[0].tick_params(direction='out', length=5, width=2,
                        colors='k', labelsize=12)
    axes[1].tick_params(direction='out', length=5, width=2,
                        colors='k', labelsize=12)
    axes[0].set_title('Actual', fontsize=12)
    axes[1].set_title('Predicted', fontsize=12)
    plt.scatter(kmeans.cluster_centers_[:, 0],
                kmeans.cluster_centers_[:,1],
                s = 25, c = 'Black', label = 'Centroids')
    plt.legend()
    plt.show()

# Output: The Elbow Method Curve wrt Inertia
K_array=np.arange(1,11,1)
plt.figure(figsize=(10,5))
plt.plot(K_array,sse)
plt.xlim(0, 10)
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("The Elbow Method Curve to find Optimum k")
plt.show()
```

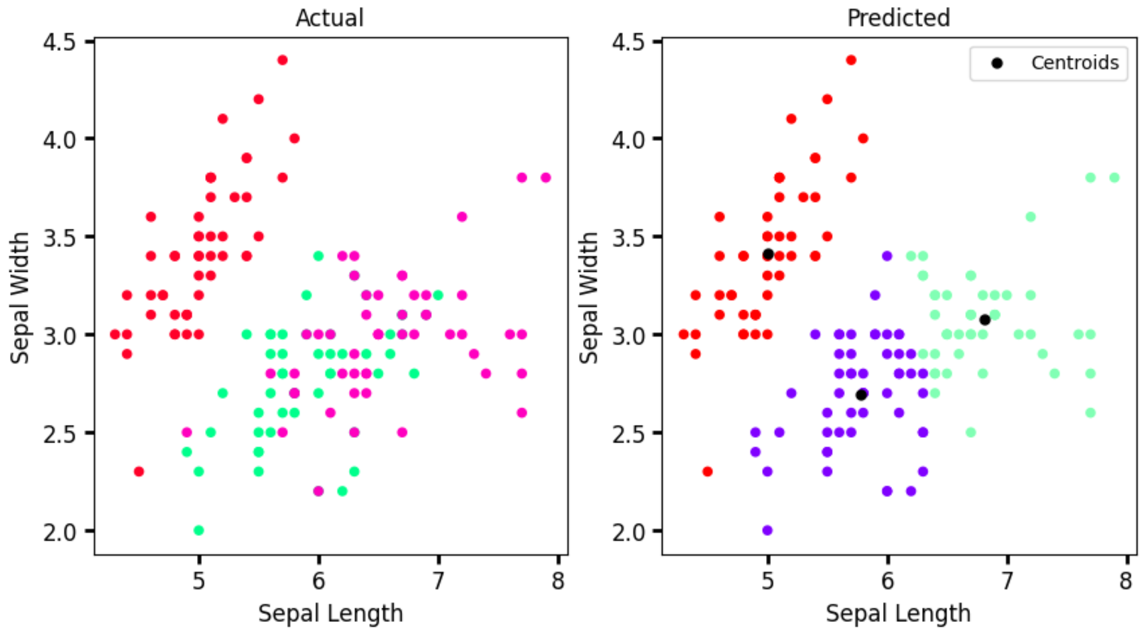
```
For k =  1

  Center  1  =  [5.84832215 3.05100671]
```



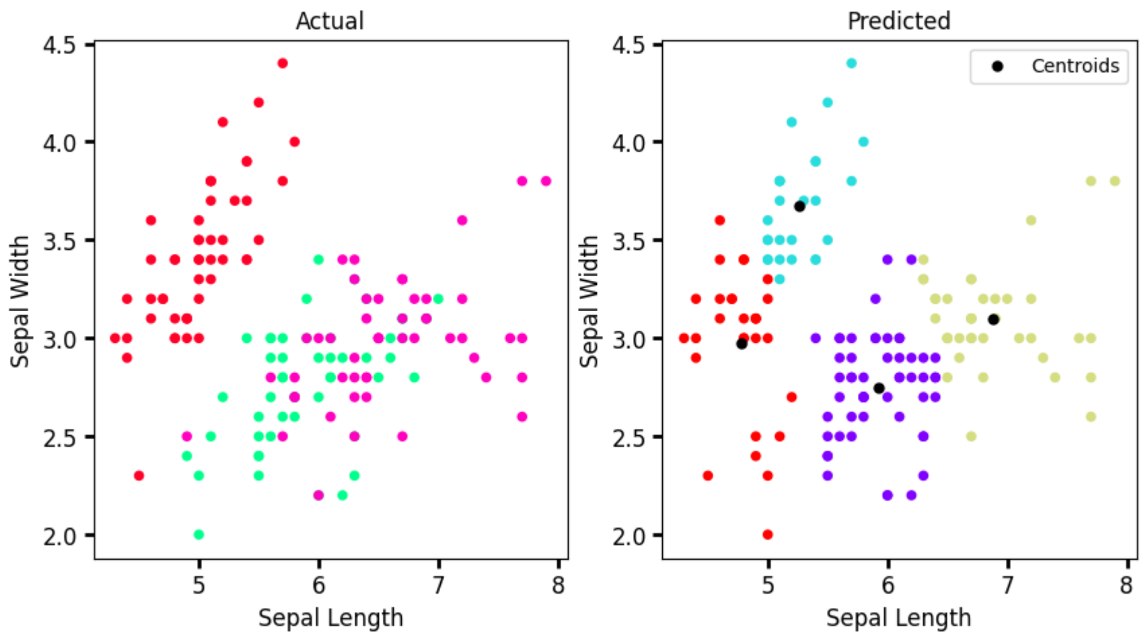

```
For k =  2

  Center  1  =  [5.22560976 3.12073171]
  Center  2  =  [6.61044776 2.96567164]
```



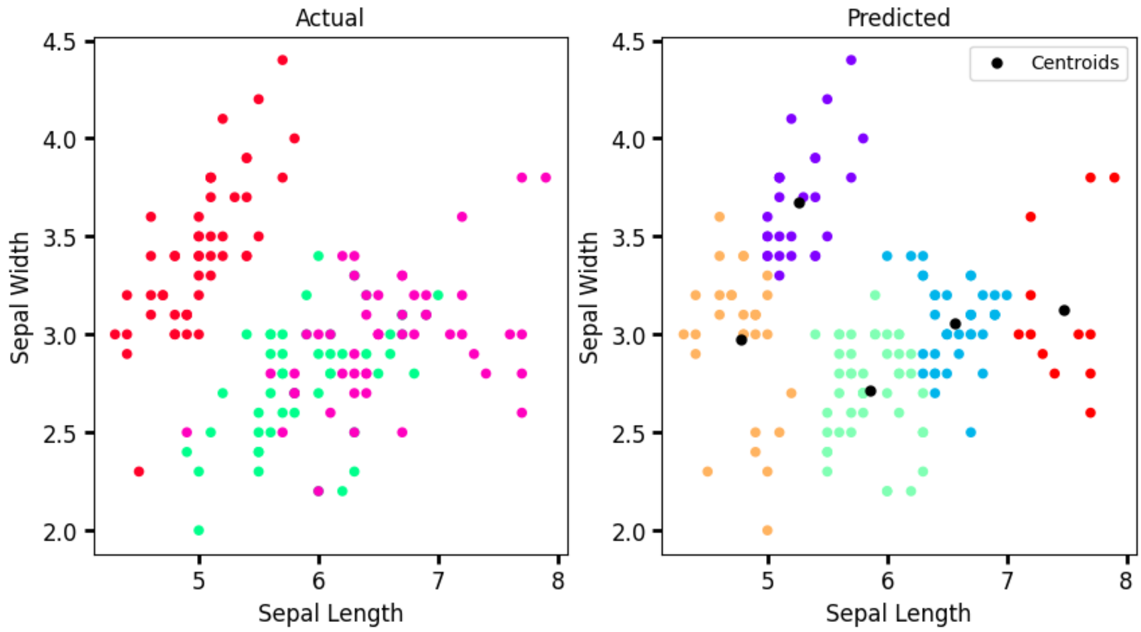

```
For k =  3

  Center  1  =  [5.77358491 2.69245283]
  Center  2  =  [6.81276596 3.07446809]
  Center  3  =  [5.00408163 3.41632653]
```



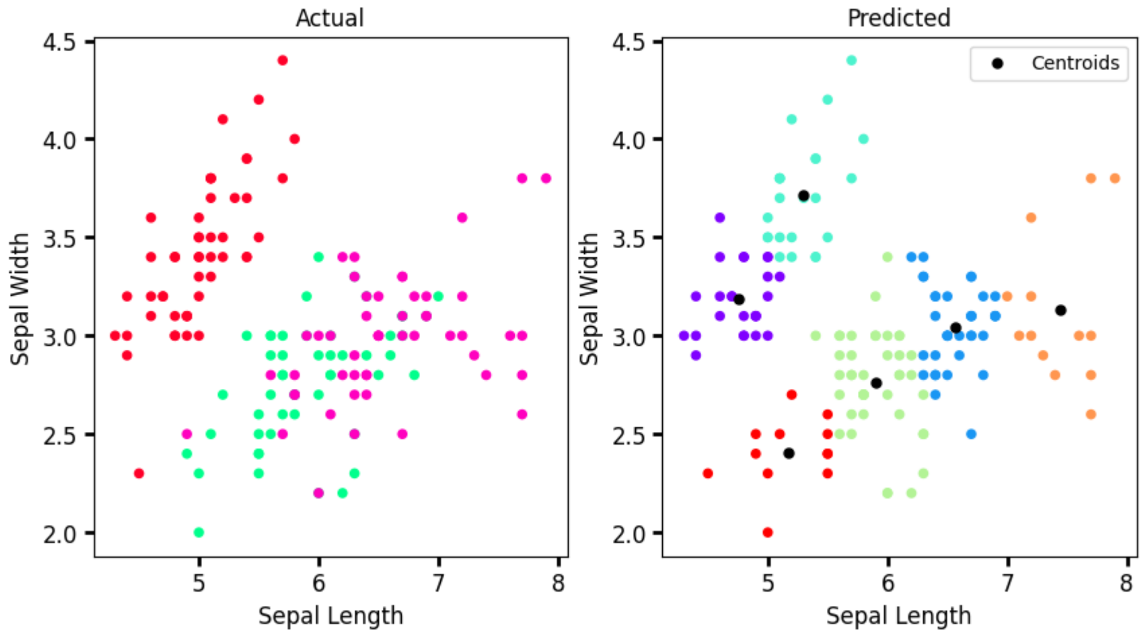

```
For k =  4

  Center  1  =  [5.9245283 2.7509434]
  Center  2  =  [5.26153846 3.67692308]
  Center  3  =  [6.8804878  3.09756098]
  Center  4  =  [4.77586207 2.97241379]
```



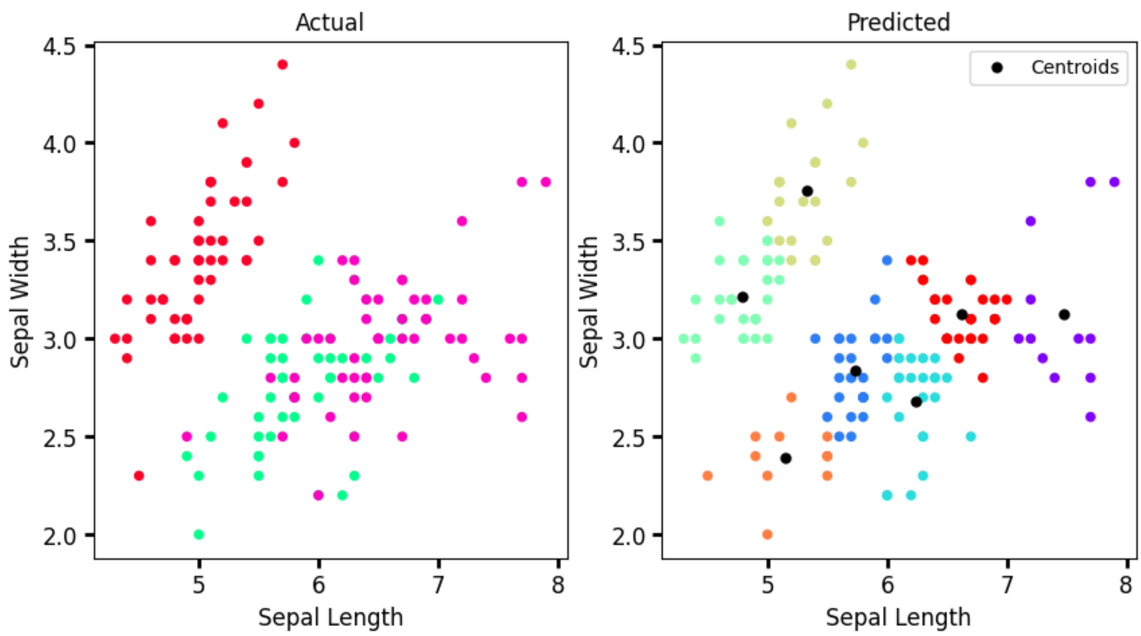

```
For k =  5

  Center  1  =  [5.26153846 3.67692308]
  Center  2  =  [6.56216216 3.05945946]
  Center  3  =  [5.85777778 2.71333333]
  Center  4  =  [4.77586207 2.97241379]
  Center  5  =  [7.475 3.125]
```



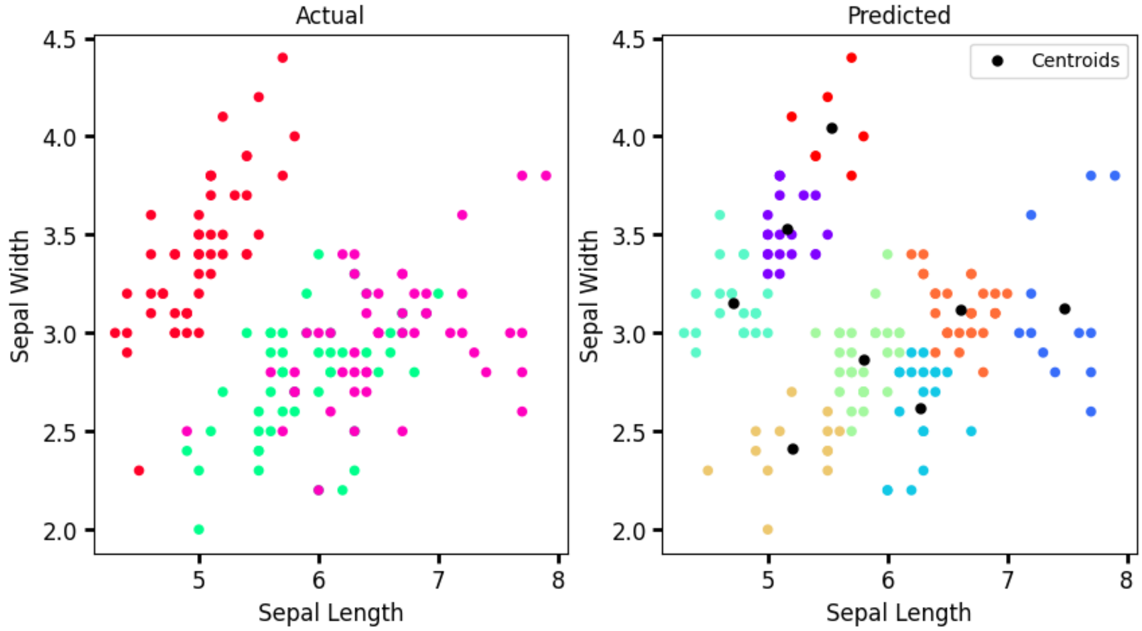

```
For k =  6

  Center  1  =  [4.76   3.184]
  Center  2  =  [6.56571429 3.04571429]
  Center  3  =  [5.29130435 3.7173913 ]
  Center  4  =  [5.90487805 2.76341463]
  Center  5  =  [7.43846154 3.13076923]
  Center  6  =  [5.175      2.40833333]
```



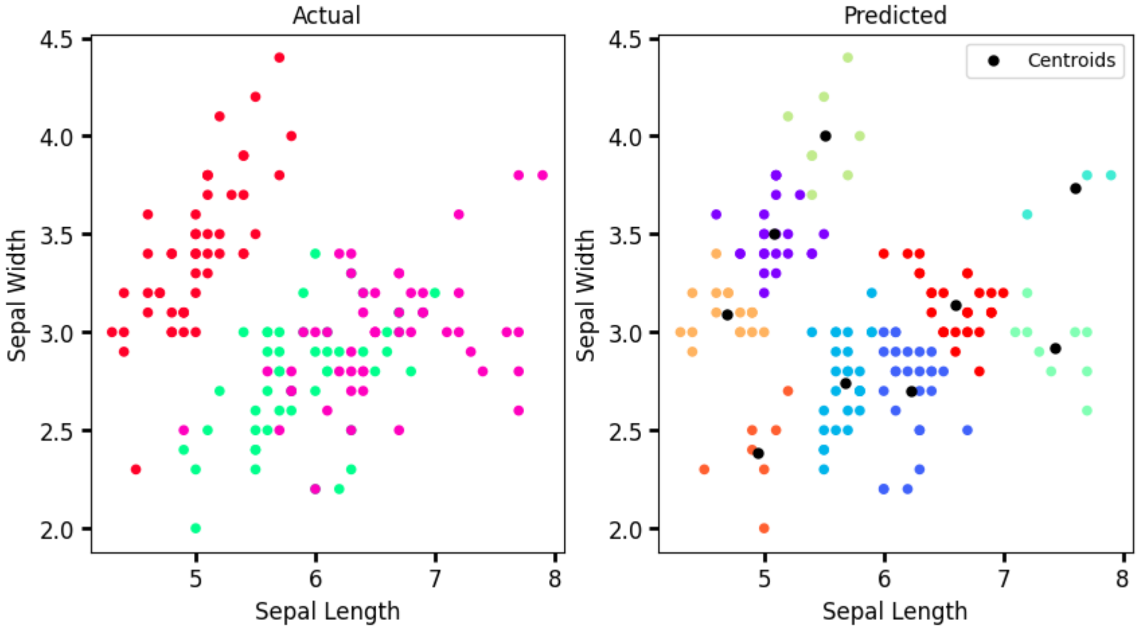

```
For k =  7

  Center  1  =  [7.475 3.125]
  Center  2  =  [5.73846154 2.83846154]
  Center  3  =  [6.24166667 2.67916667]
  Center  4  =  [4.78928571 3.21428571]
  Center  5  =  [5.33   3.755]
  Center  6  =  [5.14545455 2.39090909]
  Center  7  =  [6.62142857 3.12857143]
```



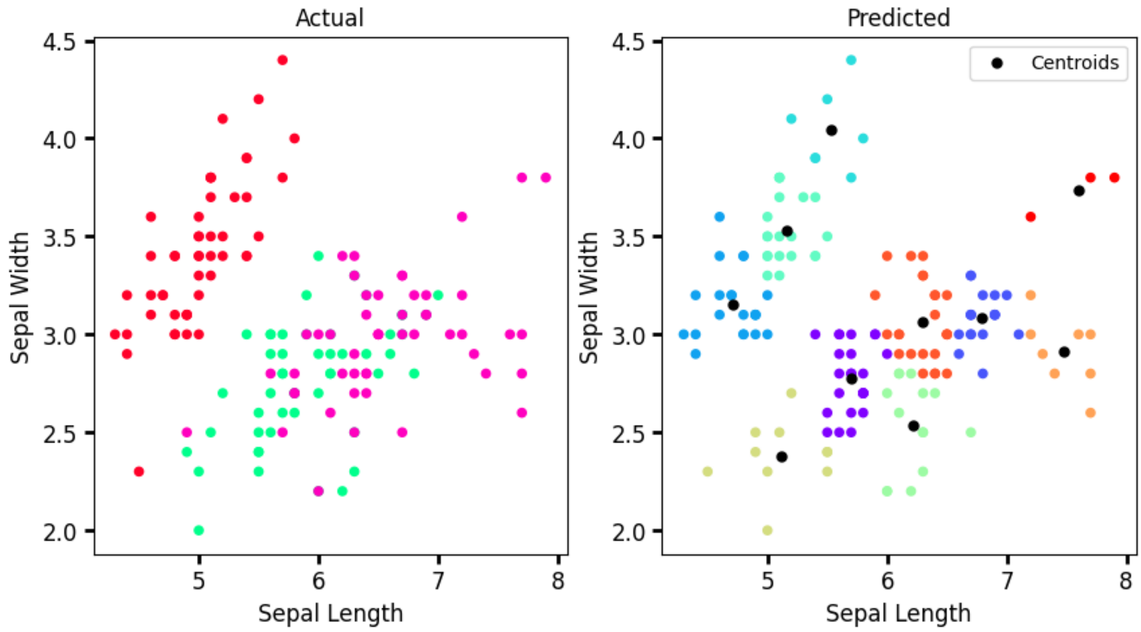

```
For k =  8

  Center  1  =  [5.155 3.53 ]
  Center  2  =  [7.475 3.125]
  Center  3  =  [6.26842105 2.62105263]
  Center  4  =  [4.70952381 3.15238095]
  Center  5  =  [5.8        2.86785714]
  Center  6  =  [5.20769231 2.41538462]
  Center  7  =  [6.6137931  3.12068966]
  Center  8  =  [5.52857143 4.04285714]
```

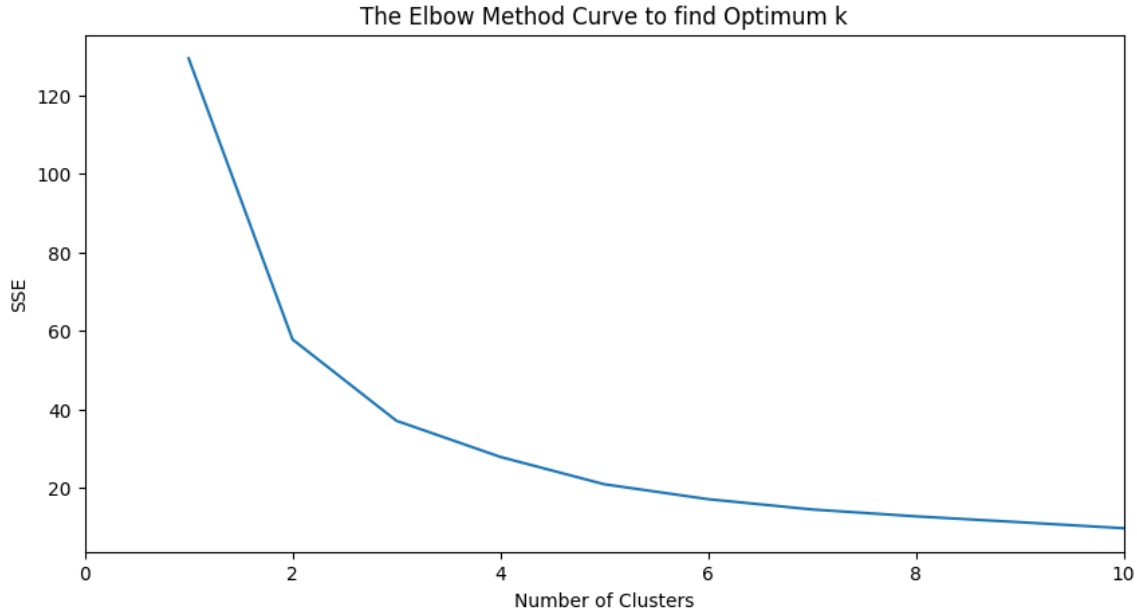


```
For k =  9

  Center  1  =  [5.0826087 3.5       ]
  Center  2  =  [6.22307692 2.7        ]
  Center  3  =  [5.67407407 2.74444444]
  Center  4  =  [7.6        3.73333333]
  Center  5  =  [7.43333333 2.92222222]
  Center  6  =  [5.5125 4.    ]
  Center  7  =  [4.68823529 3.09411765]
  Center  8  =  [4.94285714 2.38571429]
  Center  9  =  [6.6        3.13793103]
```




```
For k =  10

  Center  1  =  [5.7        2.77916667]
  Center  2  =  [6.78888889 3.08333333]
  Center  3  =  [4.70952381 3.15238095]
  Center  4  =  [5.52857143 4.04285714]
  Center  5  =  [5.155 3.53 ]
  Center  6  =  [6.21428571 2.53571429]
  Center  7  =  [5.11 2.38]
  Center  8  =  [7.475  2.9125]
  Center  9  =  [6.29166667 3.0625    ]
  Center  10  =  [7.6        3.73333333]
```



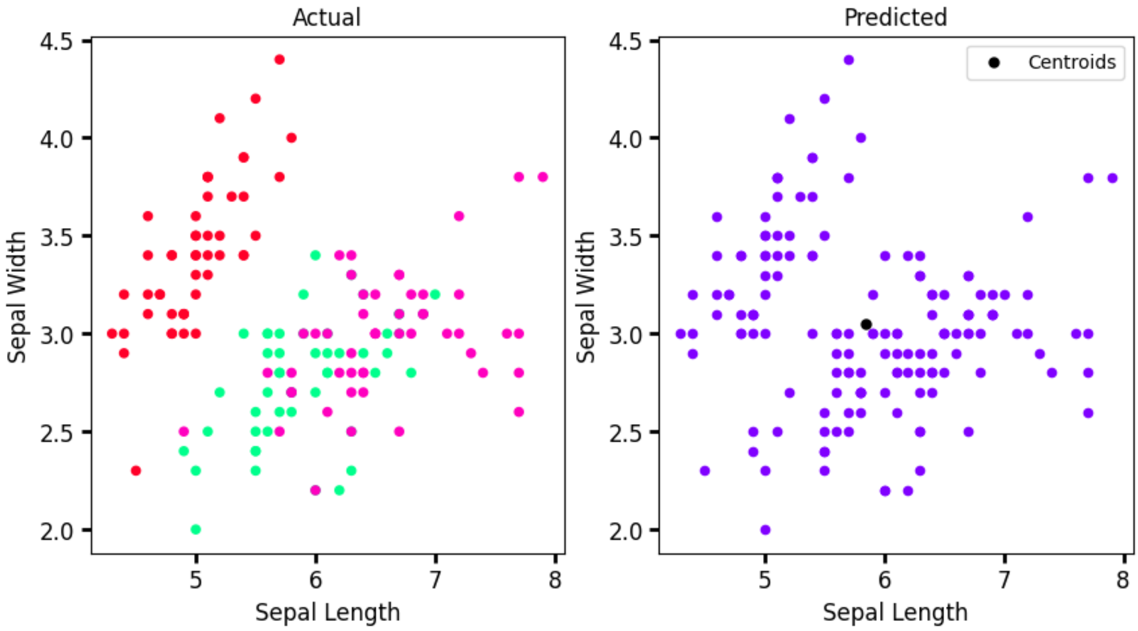


Python Program for K Means Clustering (Manual).

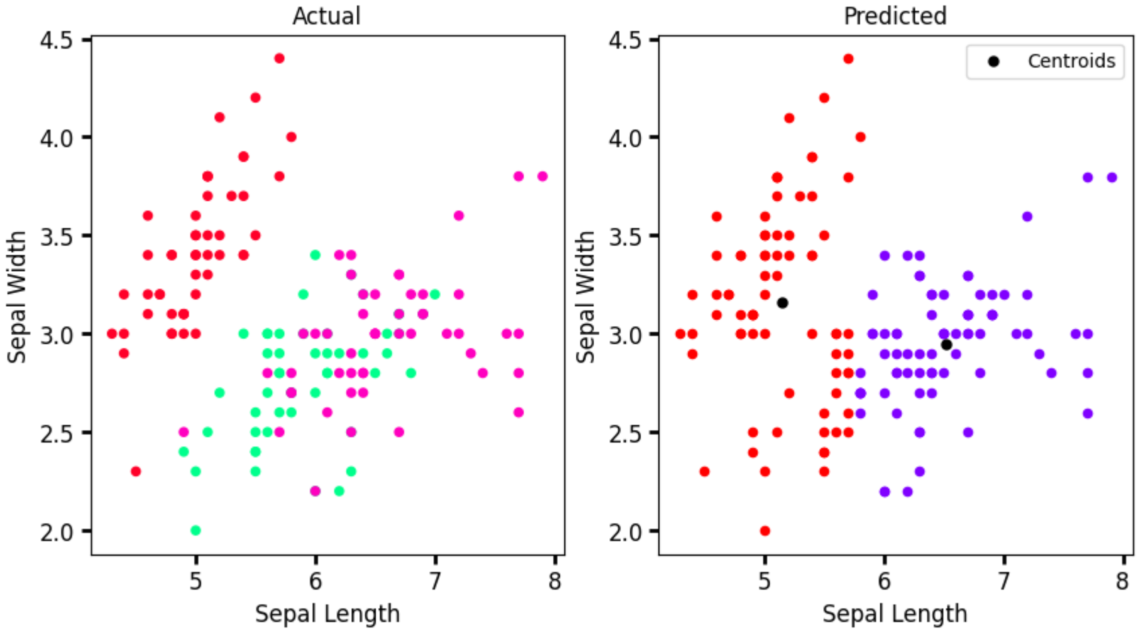
In [ ]:
```
# Input: Dataset
dataset = pd.read_csv('iris.data')

# Taking only two attributes of the Dataset
X = dataset.iloc[:, [0, 1]].values

m=X.shape[0] # Number of Training Examples
n=X.shape[1] # Number of Features
n_iter=1000

# K Means Clustering Algorithm
WCSS_array=[]
for K in range(1,11):            # Model training for different k values
    Centroids=np.array([]).reshape(n,0)
    for i in range(K):
        rand=random.randint(0,m-1)
        # Taking Random Samples as Initial Centroids
        Centroids=np.c_[Centroids,X[rand]]
    Output={}
```

```
    for i in range(n_iter):
        EuclidianDistance=np.array([]).reshape(m,0)
        for k in range(K):
            tempDist=np.sum((X-Centroids[:,k])**2,axis=1)
            # Calculating Euclidean Distance
            EuclidianDistance=np.c_[EuclidianDistance,tempDist]
        C=np.argmin(EuclidianDistance,axis=1)+1
        Y={}
        for k in range(K):
            Y[k+1]=np.array([]).reshape(2,0)
        for i in range(m):
            Y[C[i]]=np.c_[Y[C[i]],X[i]]
        for k in range(K):
            Y[k+1]=Y[k+1].T
        for k in range(K):
            Centroids[:,k]=np.mean(Y[k+1],axis=0)
        Output=Y                # Final Centers
    ys = [str(i) for i in range(K)]
    colors = cm.rainbow(np.linspace(0, 1, len(ys)))

    # Output: The Cluster Centers obtained
    print("\n\nFor K = ",K)
    for k in range(K):
        print("  Center ",k+1," = ",' '.join(map(str, Centroids[:,k])))

    # Output: The Actual Clusters vs Predicted Clusters for given value c
    fig, axes = plt.subplots(1, 2, figsize=(10,5))
    axes[0].scatter(test_x[:, 0], test_x[:, 1],
                    c=actuallabel, cmap='gist_rainbow', s=20)
    for k in range(K):
        axes[1].scatter(Output[k+1][:,0],
                        Output[k+1][:,1], color=colors[k], s=20)
    axes[0].set_xlabel('Sepal Length', fontsize=12)
    axes[0].set_ylabel('Sepal Width', fontsize=12)
    axes[1].set_xlabel('Sepal Length', fontsize=12)
    axes[1].set_ylabel('Sepal Width', fontsize=12)
    axes[0].tick_params(direction='out', length=5, width=2, colors='k', l
    axes[1].tick_params(direction='out', length=5, width=2, colors='k', l
    axes[0].set_title('Actual', fontsize=12)
    axes[1].set_title('Predicted', fontsize=12)
    plt.scatter(Centroids[0,:],Centroids[1,:],s=25,c='Black',label='Centr
    plt.legend()
    plt.show()

    # Calculating the sum of distances of samples to their closest cluste
    wcss=0
    for k in range(K):
        wcss+=np.sum((Output[k+1]-Centroids[:,k])**2)
    WCSS_array.append(wcss)

# Output: The Elbow Method Curve wrt Inertia
K_array=np.arange(1,11,1)
plt.figure(figsize=(10,5))
plt.plot(K_array,WCSS_array)
plt.xlim(0, 10)
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("The Elbow Method Curve to find Optimum k")
plt.show()
```

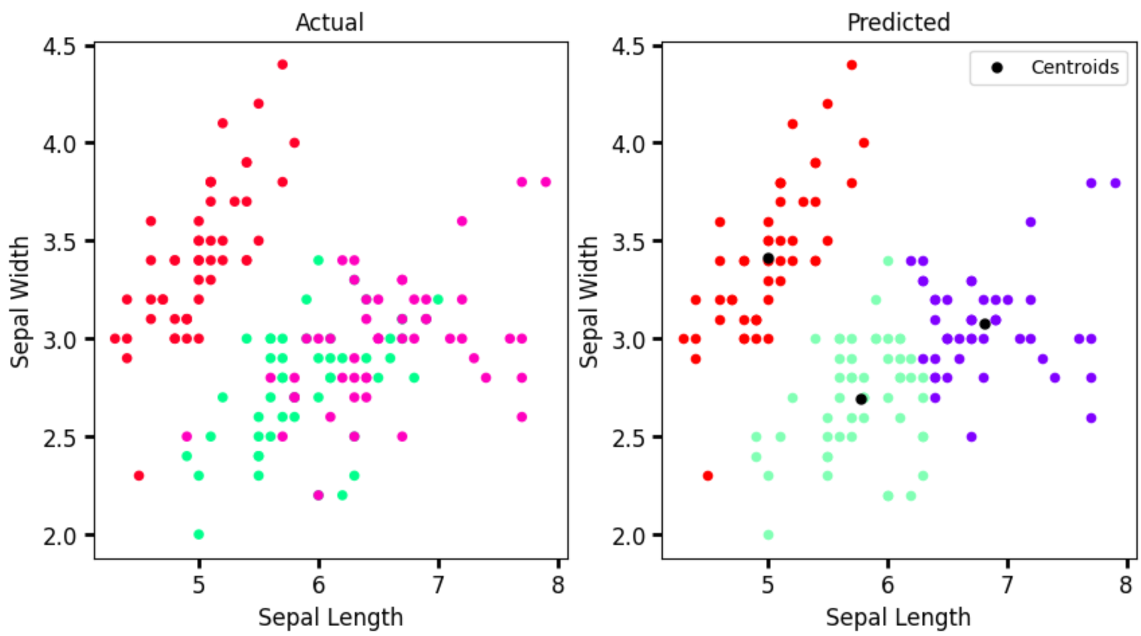
```
For K =  1
  Center  1  =  5.8483221476510066 3.051006711409396
```



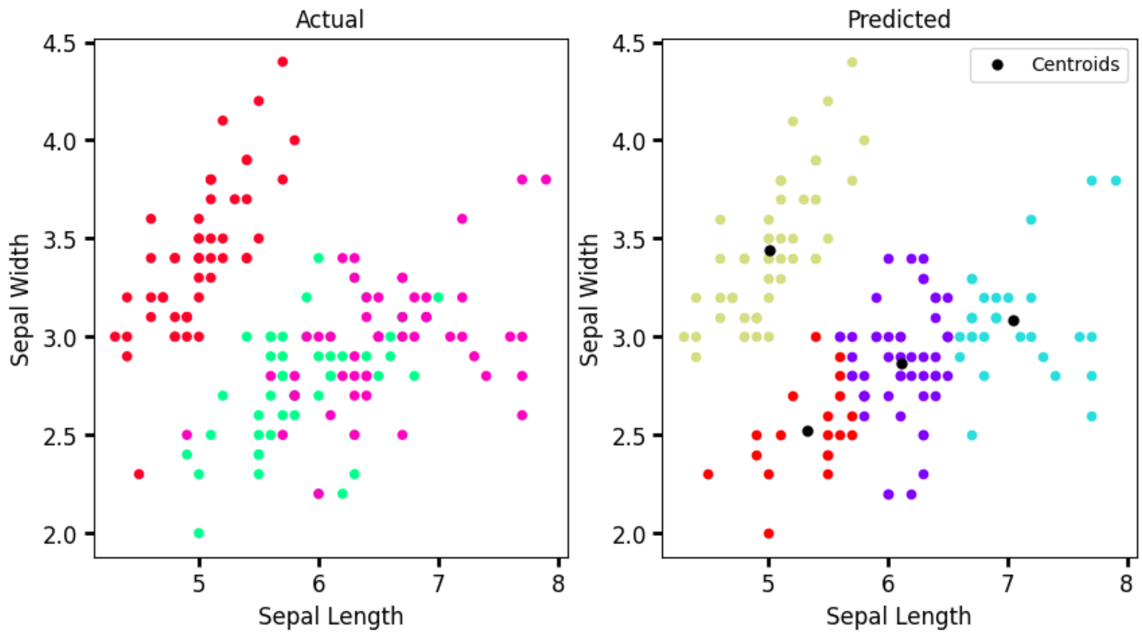

```
For K =  2
  Center  1  =  6.518421052631578 2.948684210526316
  Center  2  =  5.15068493150685 3.1575342465753424
```



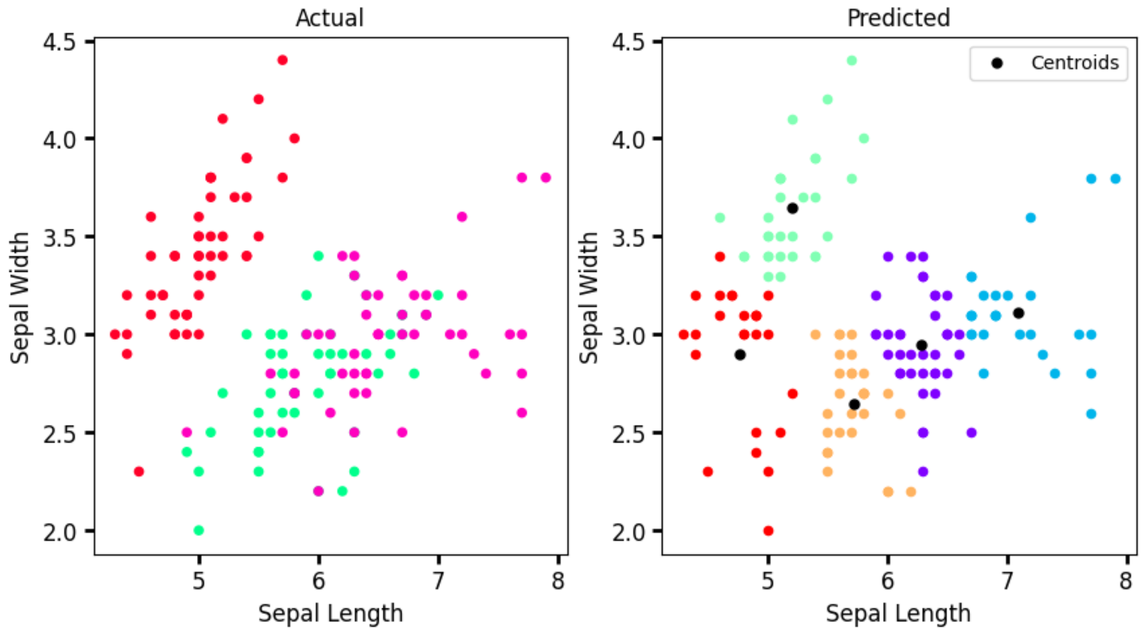

```
For K =  3
  Center  1  =  6.812765957446807 3.074468085106383
  Center  2  =  5.773584905660377 2.692452830188679
  Center  3  =  5.004081632653061 3.4163265306122454
```



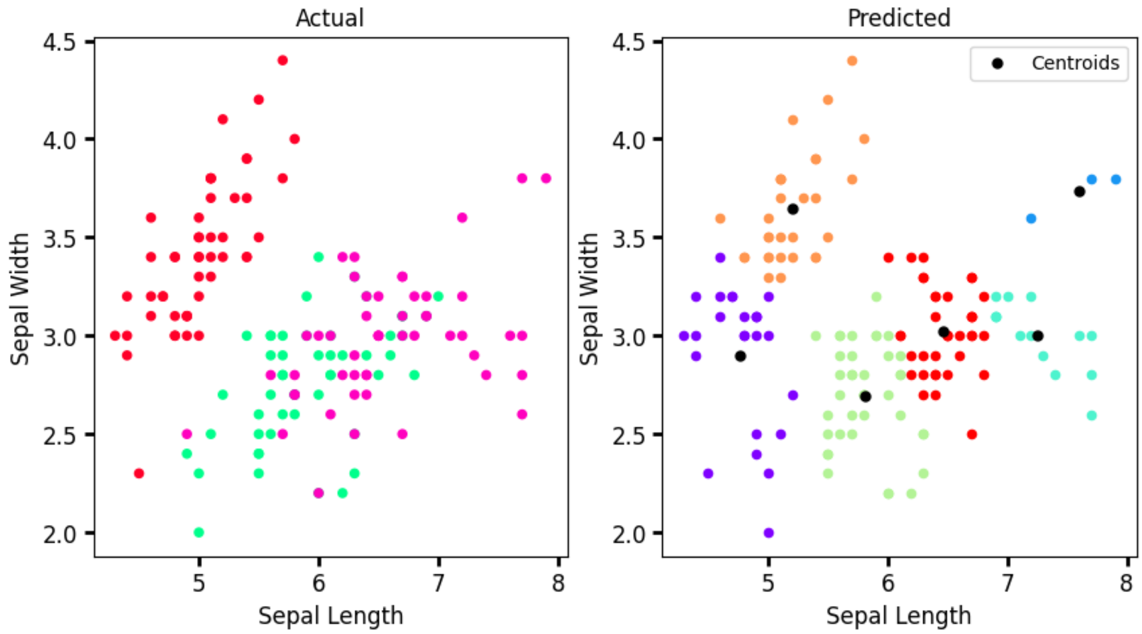

```
For K =  4
  Center  1  =  6.113461538461538 2.8673076923076923
  Center  2  =  7.05 3.0833333333333326
  Center  3  =  5.014583333333333 3.4395833333333337
  Center  4  =  5.3315789473684205 2.5210526315789474
```



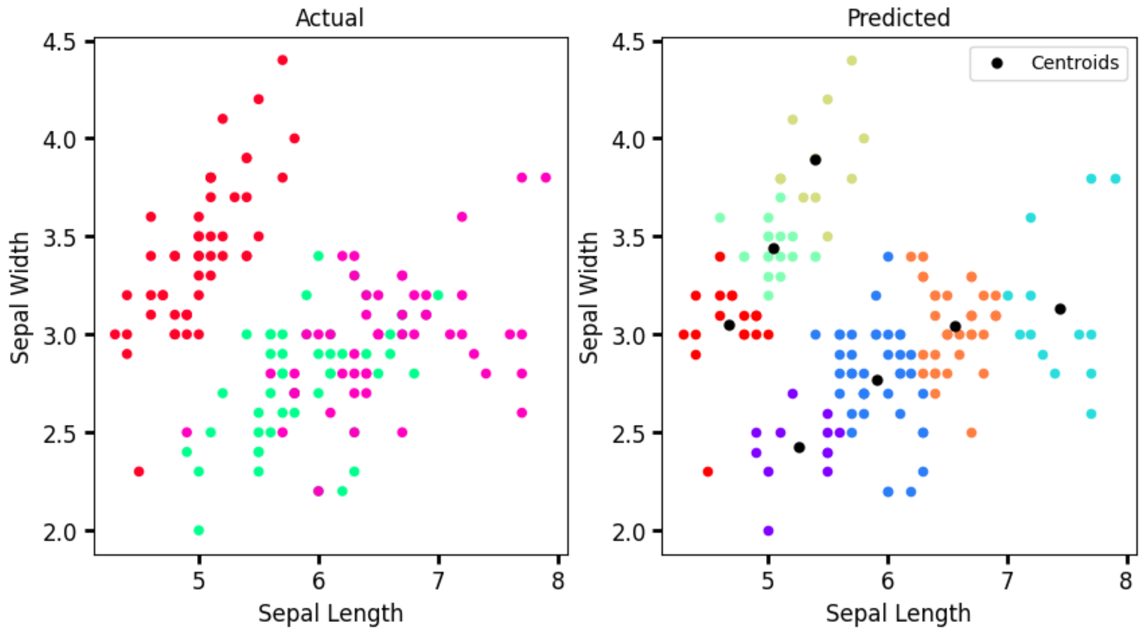

```
For K =  5
  Center  1  =  6.281578947368422 2.944736842105263
  Center  2  =  7.096296296296296 3.1148148148148147
  Center  3  =  5.2 3.6433333333333326
  Center  4  =  5.717241379310345 2.6482758620689655
  Center  5  =  4.772 2.9
```



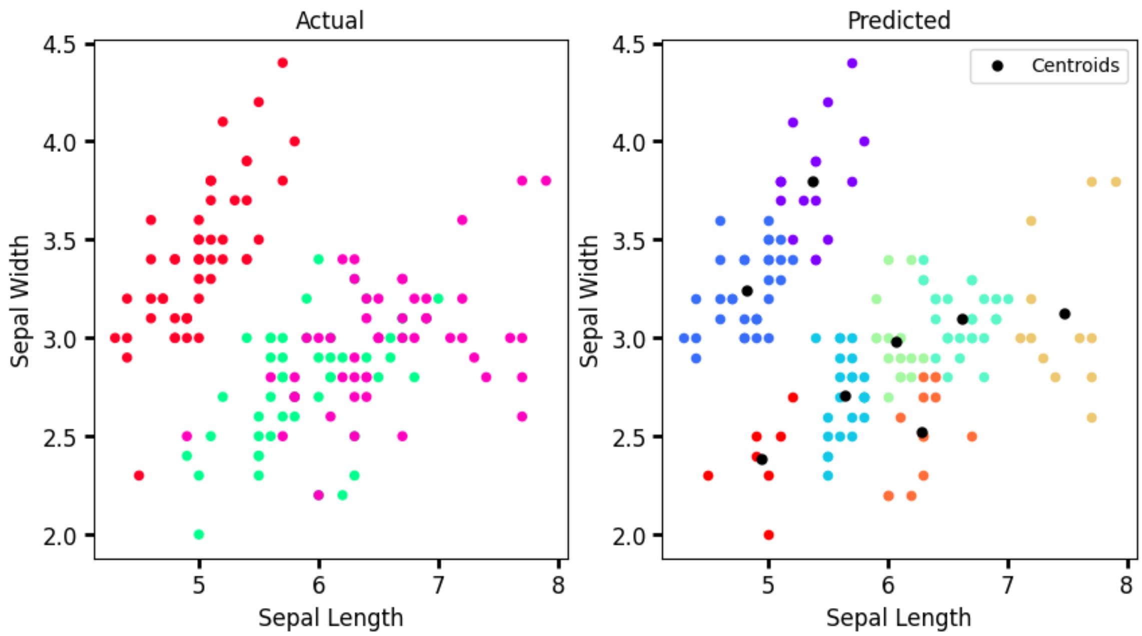

```
For K =  6
  Center  1  =  4.772 2.9
  Center  2  =  7.6000000000000005 3.7333333333333333
  Center  3  =  7.250000000000002 3.0
  Center  4  =  5.8175000000000001 2.6925
  Center  5  =  5.2 3.6433333333333326
  Center  6  =  6.462162162162161 3.0243243243243243
```



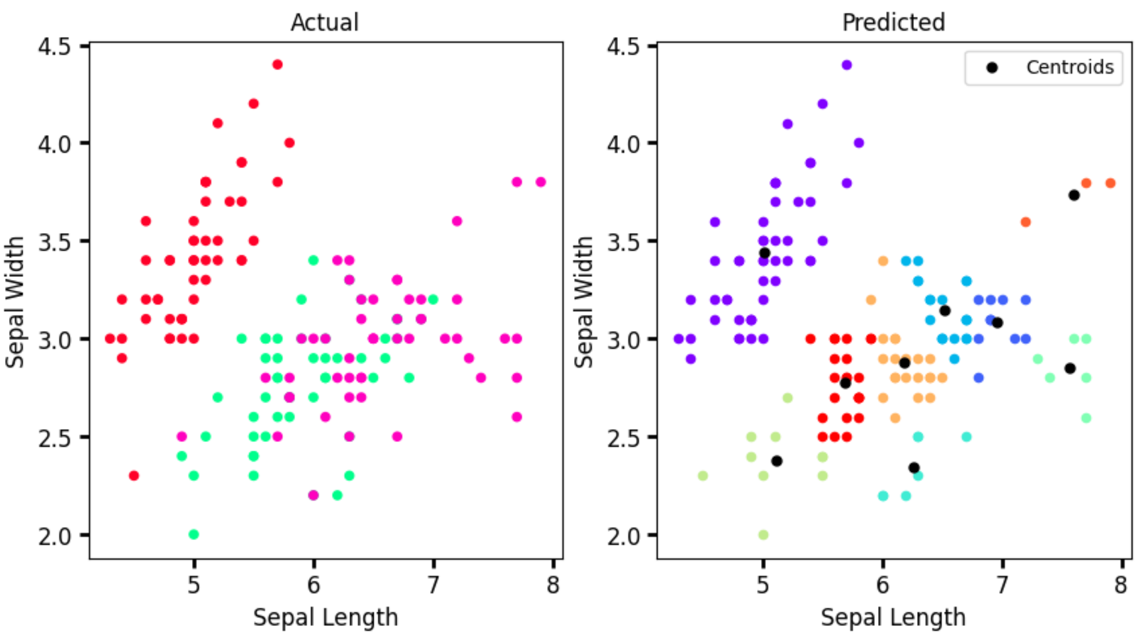

```
For K =  7
  Center  1  =  5.266666666666667 2.4250000000000003
  Center  2  =  5.9125 2.77
  Center  3  =  7.43846153846154 3.1307692307692303
  Center  4  =  5.044444444444444 3.4388888888888887
  Center  5  =  5.4 3.892307692307692
  Center  6  =  6.565714285714285 3.0457142857142854
  Center  7  =  4.677777777777778 3.0500000000000003
```



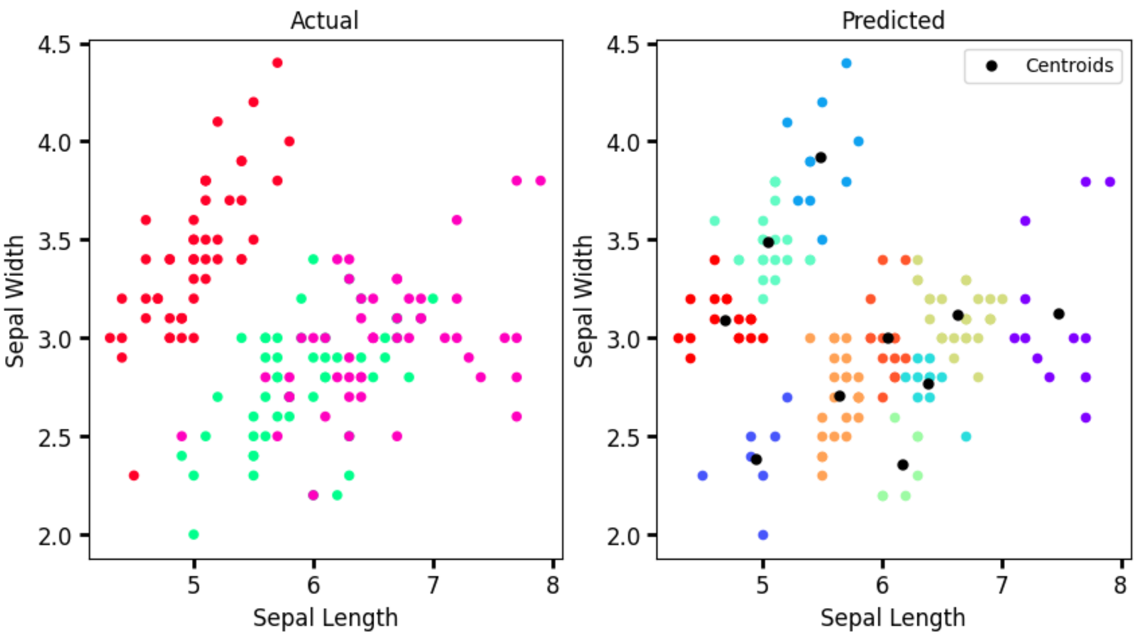

```
For K =  8
  Center  1  =  5.370588235294117 3.8
  Center  2  =  4.8193548387096765 3.2419354838709684
  Center  3  =  5.645833333333333 2.7041666666666667
  Center  4  =  6.624137931034483 3.0999999999999996
  Center  5  =  6.06875 2.98125
  Center  6  =  7.4750000000000001 3.125
  Center  7  =  6.284615384615384 2.5230769230769234
  Center  8  =  4.942857142857143 2.3857142857142857
```

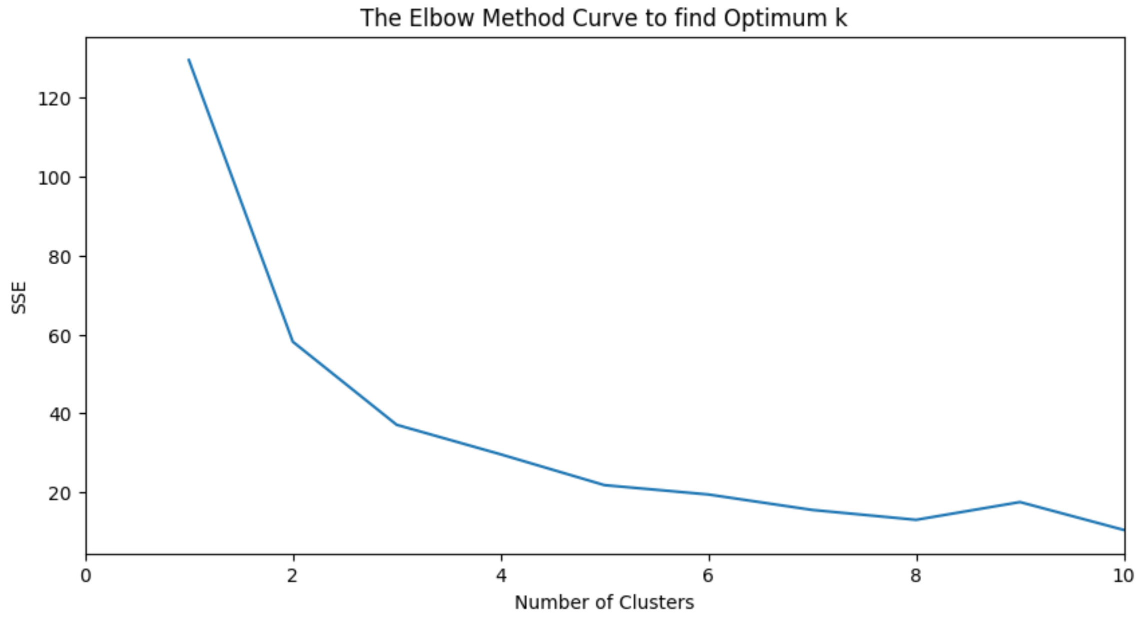



```
For K =  9
  Center  1  =  5.014583333333333 3.4395833333333337
  Center  2  =  6.954545454545454 3.0818181818181825
  Center  3  =  6.5200000000000005 3.145
  Center  4  =  6.257142857142857 2.3428571428571425
  Center  5  =  7.566666666666666 2.85
  Center  6  =  5.11 2.38
  Center  7  =  6.185714285714286 2.8761904761904757
  Center  8  =  7.6000000000000005 3.733333333333333
  Center  9  =  5.68695652173913 2.773913043478261
```




```
For K =  10
  Center  1  =  7.475000000000001 3.125
  Center  2  =  4.942857142857143 2.3857142857142857
  Center  3  =  5.49 3.9200000000000004
  Center  4  =  6.39 2.7700000000000005
  Center  5  =  5.052380952380952 3.4904761904761905
  Center  6  =  6.171428571428571 2.357142857142857
  Center  7  =  6.637037037037037 3.1185185185185187
  Center  8  =  5.645833333333333 2.704166666666667
  Center  9  =  6.042857142857144 2.9999999999999996
  Center  10  =  4.688235294117646 3.0941176470588236
```





Exercise-8

Comparison of Both Methods.

We trained both the models for different values of k with Maximum Number of Iteration allowed as 1000. The Sci-Kit Learn Method efficiently trained the model for all values of k under given parameters, whereas the Manual Method failed to iterate the same for k>=9.

We performed Elbow Method to find the Optimum Value of k, which can be observed from the plotted graphs. Both the methods give Optimum Results for k=3.

Overall, both methods are efficient, but, SciKit-Learn Method can be preferred over Manual Method as it runs faster and gives a result within few number of iterations.