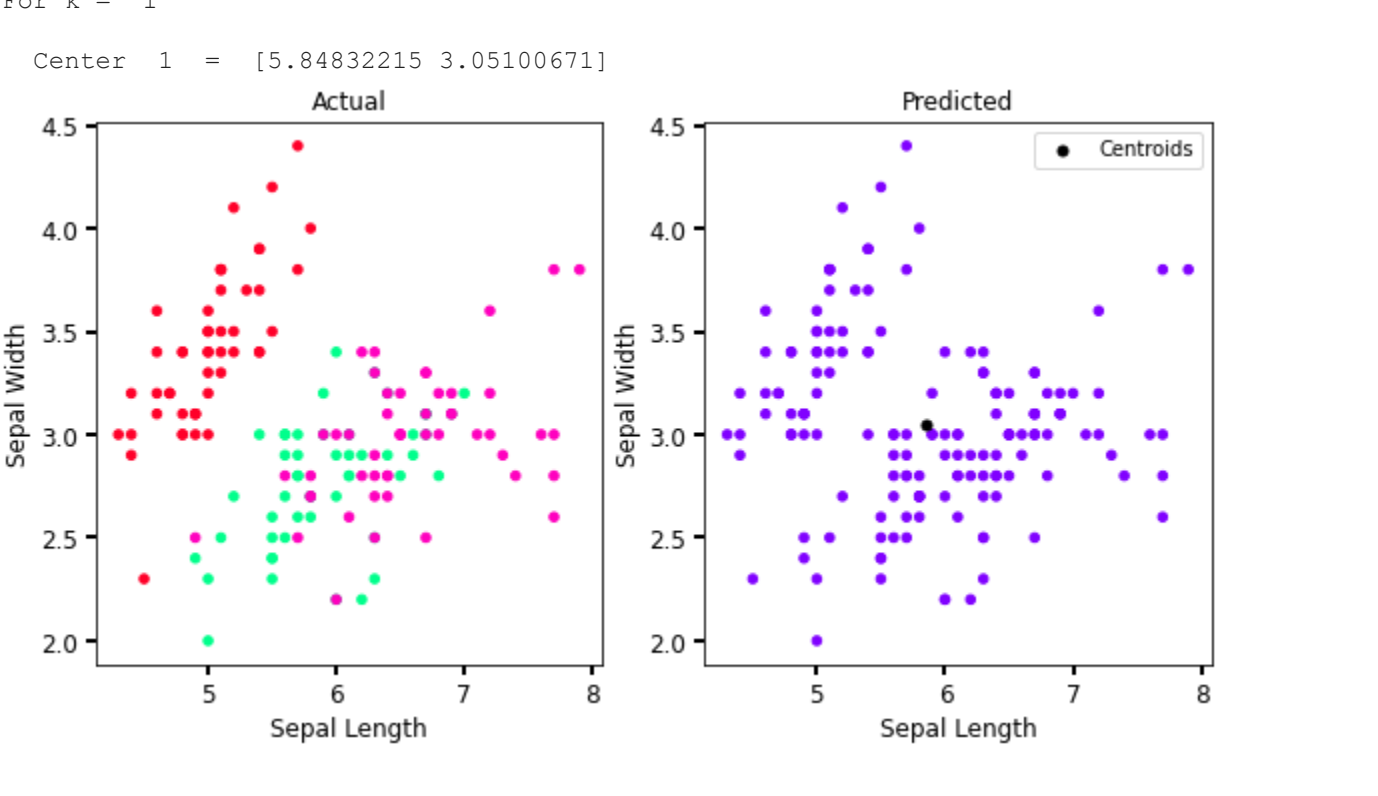
Exercise-8

Python Program for K Means Clustering (SciKit-Learn).

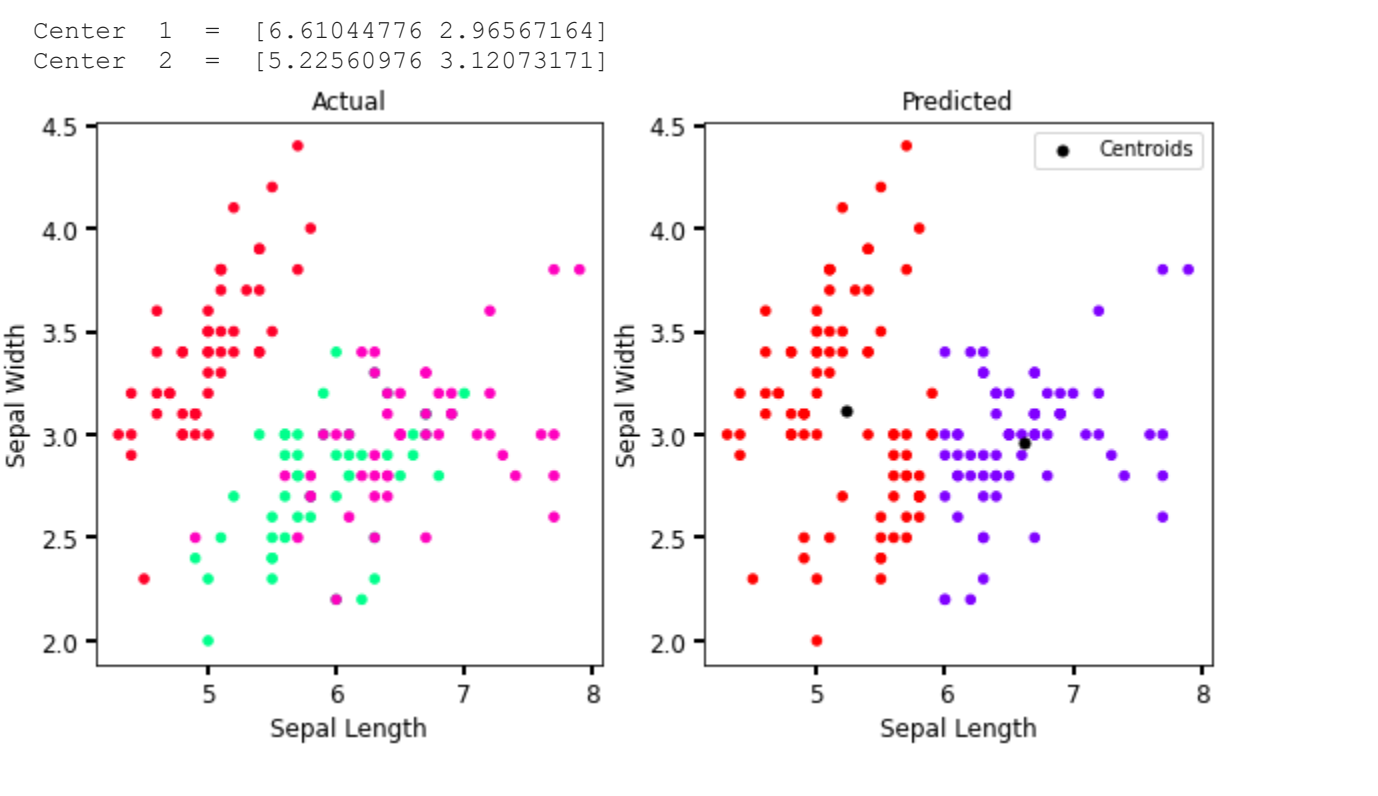
In [11]:

```python
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 values of k

    # 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 cluster center
    sse.append(kmeans.inertia_)

    # Output: The Actual Cluster Plot vs Predicted Cluster Plot for given value of k
    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()
```

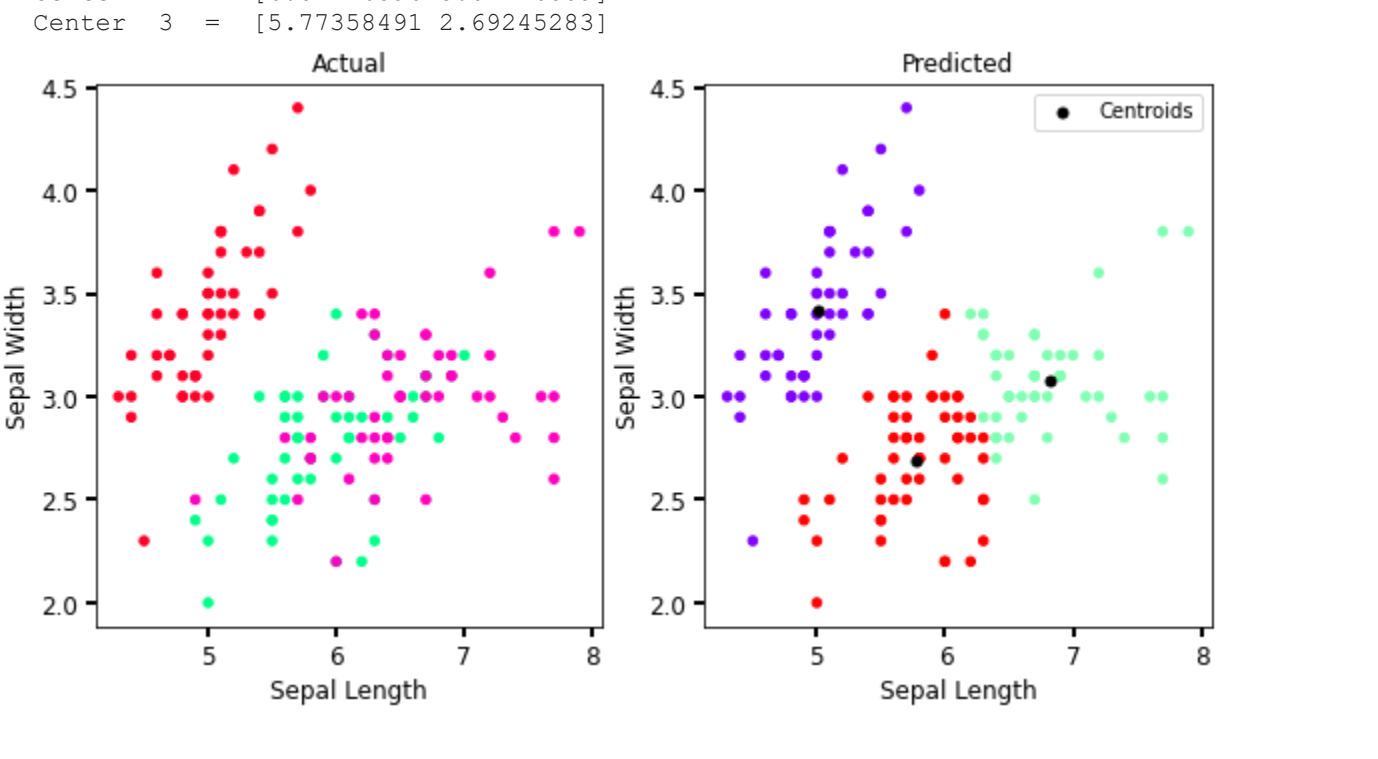
```
For k =  1

  Center  1  =  [5.84832215 3.05100671]
```



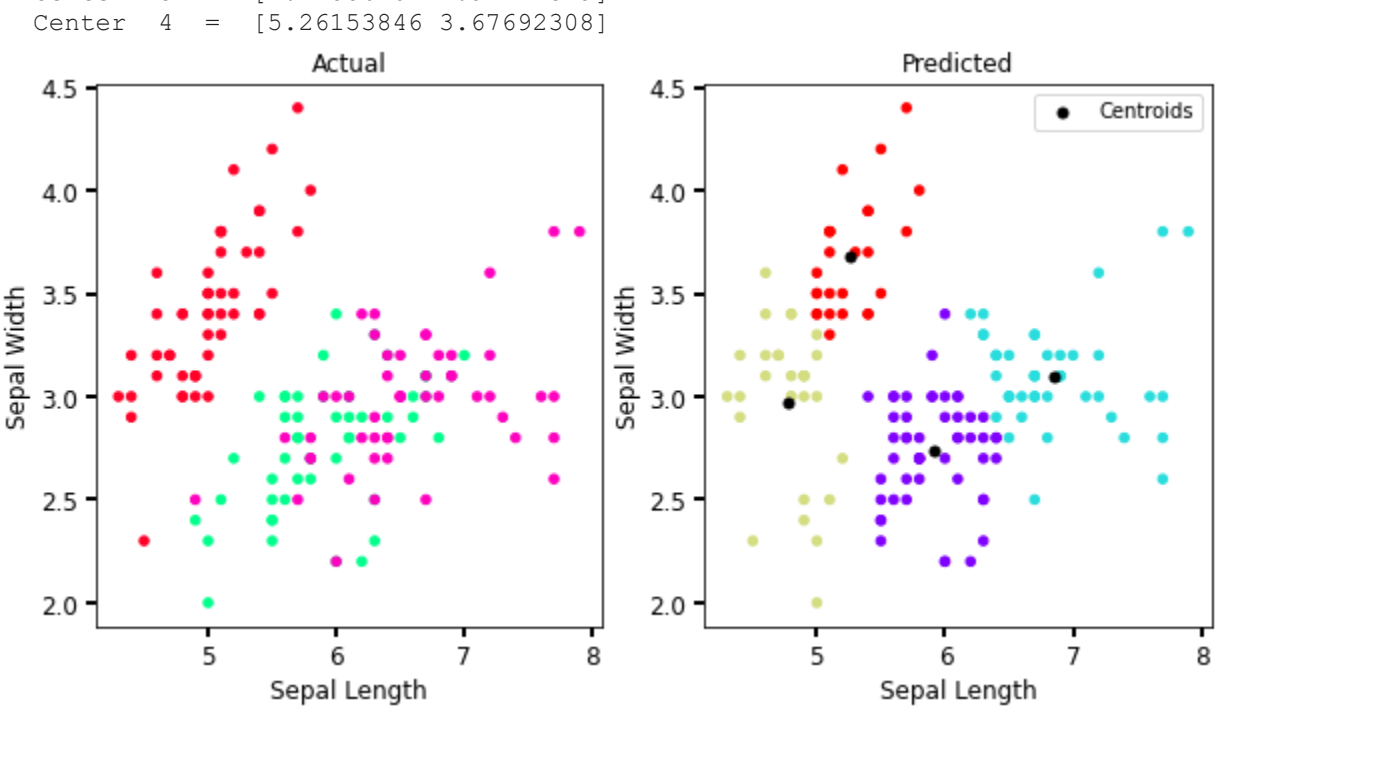
```
For k =  2

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



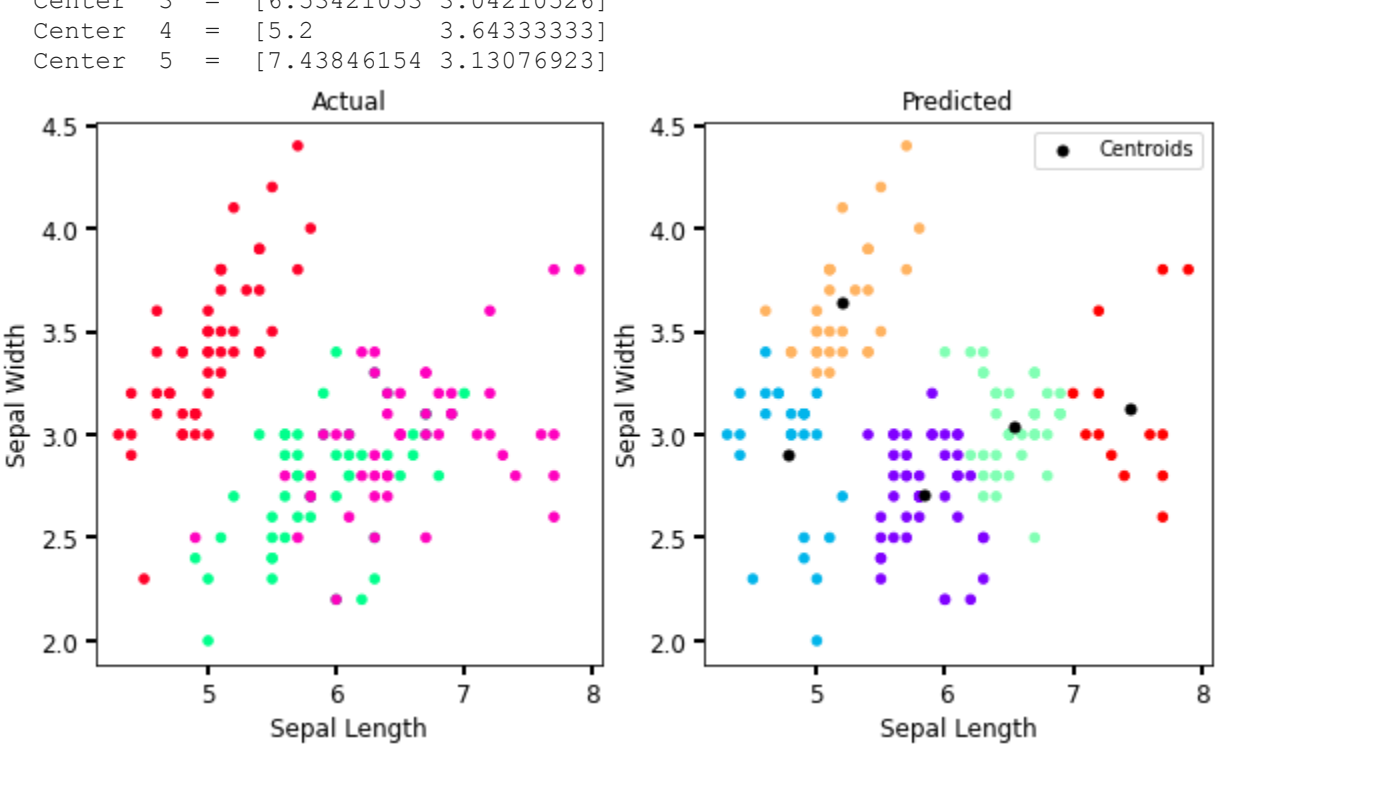
```
For k =  3

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



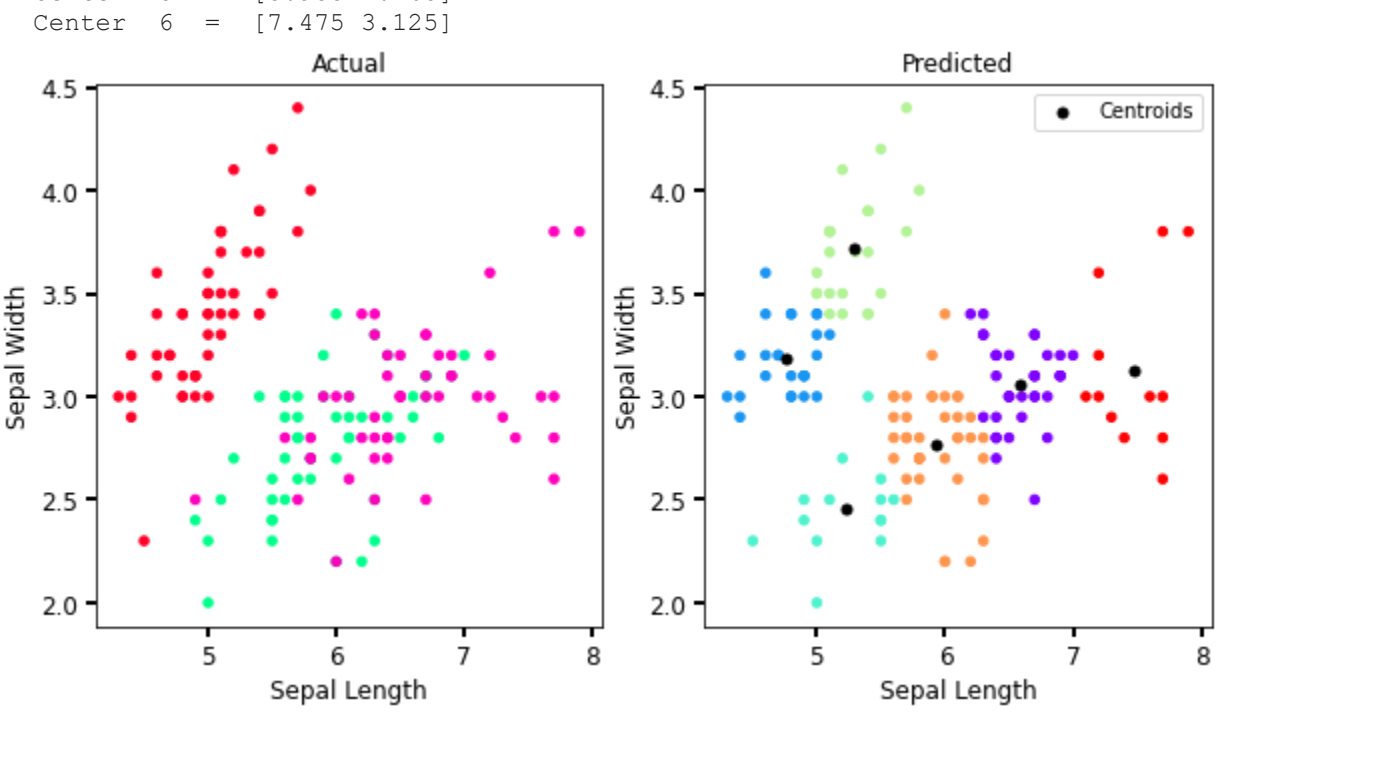
```
For k =  4

  Center  1  =  [5.90980392 2.73529412]
  Center  2  =  [6.85348837 3.1       ]
  Center  3  =  [4.77586207 2.97241379]
  Center  4  =  [5.26153846 3.67692308]
```



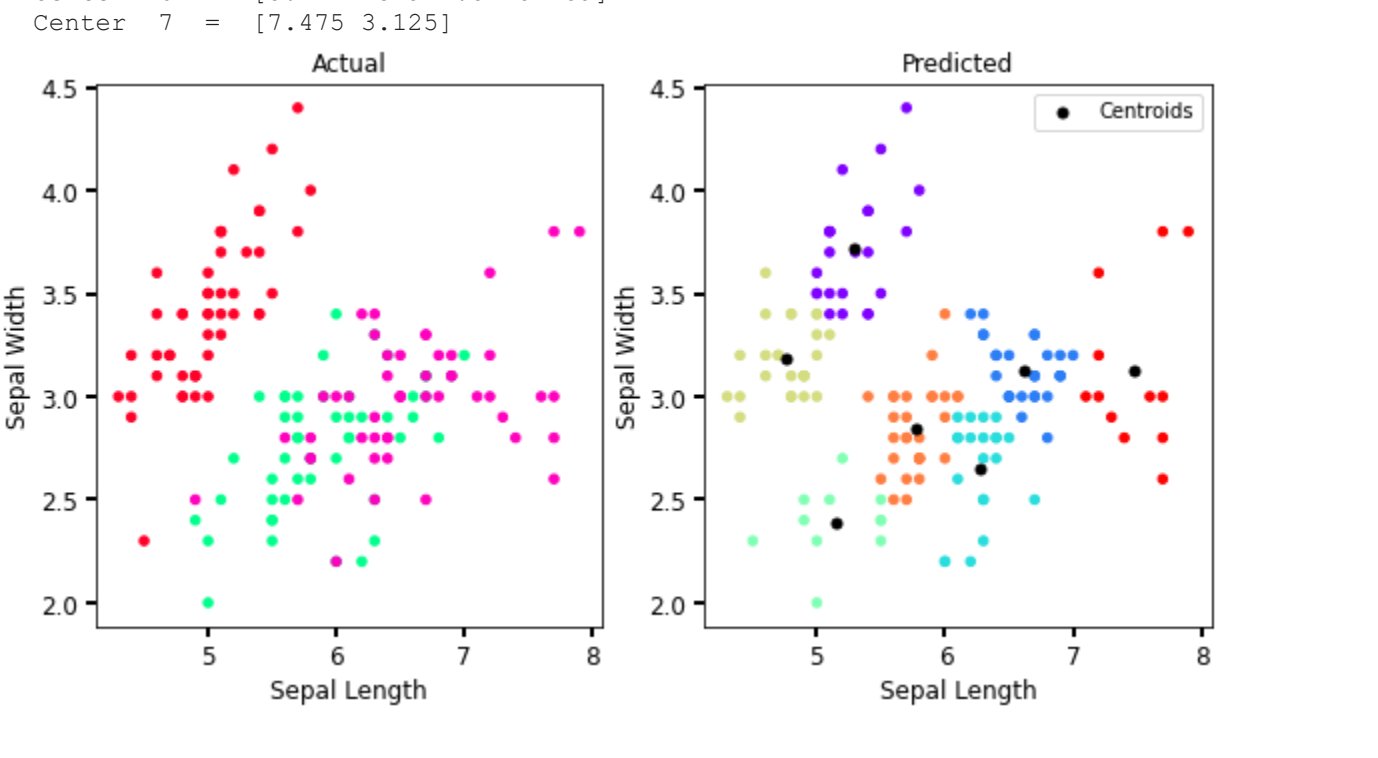
```
For k =  5

  Center  1  =  [5.83953488 2.70930233]
  Center  2  =  [4.772 2.9  ]
  Center  3  =  [6.53421053 3.04210526]
  Center  4  =  [5.2        3.64333333]
  Center  5  =  [7.43846154 3.13076923]
```



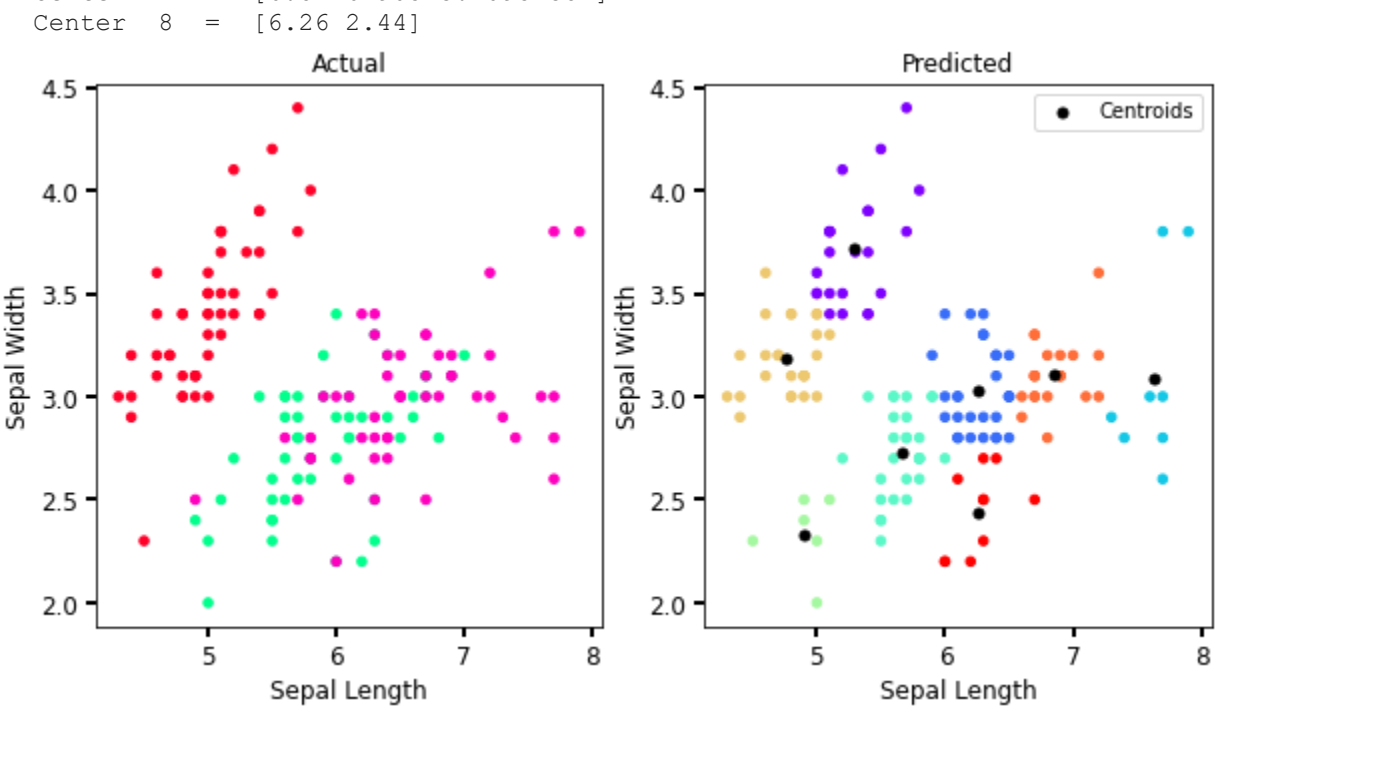
```
For k =  6

  Center  1  =  [6.58571429 3.05714286]
  Center  2  =  [4.76  3.184]
  Center  3  =  [5.22142857 2.45714286]
  Center  4  =  [5.29130435 3.7173913 ]
  Center  5  =  [5.935 2.765]
  Center  6  =  [7.475 3.125]
```



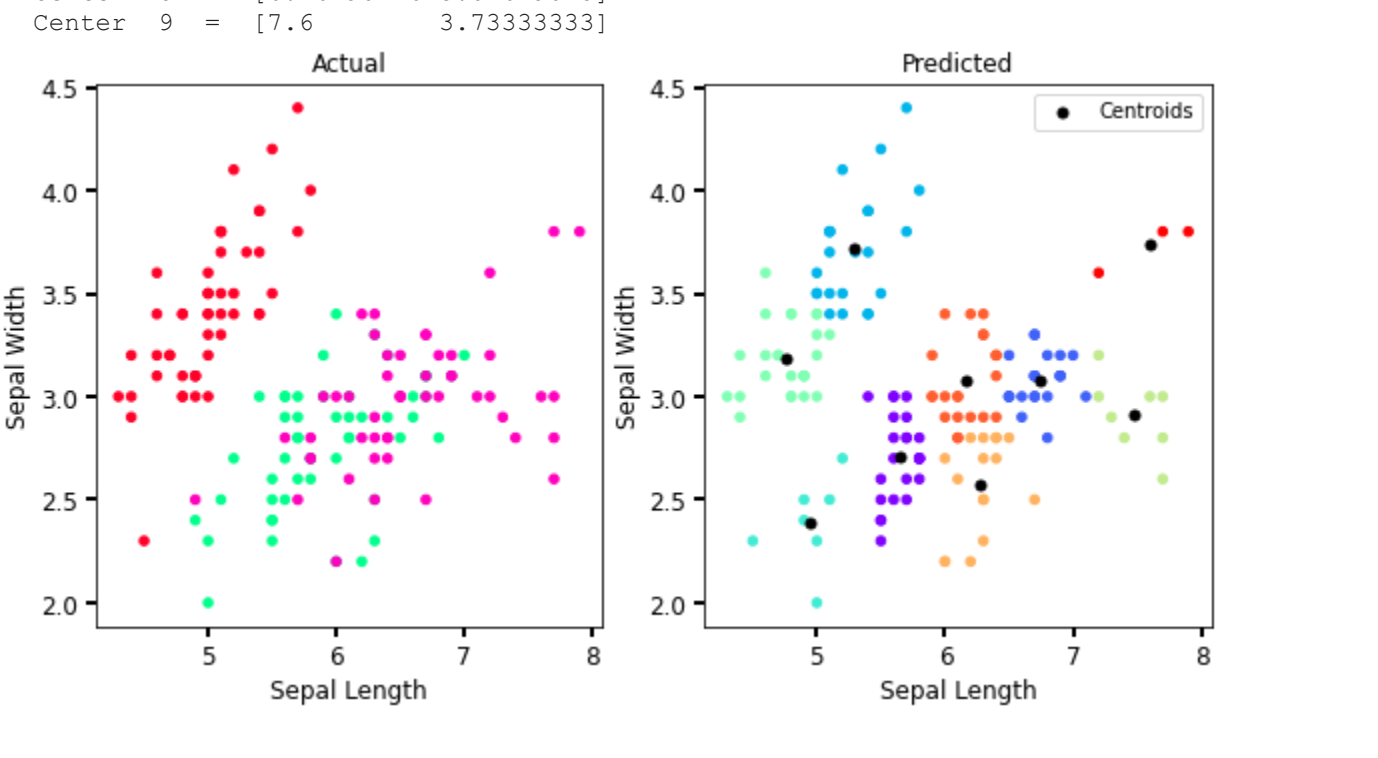
```
For k =  7

  Center  1  =  [5.29130435 3.7173913 ]
  Center  2  =  [6.62142857 3.12857143]
  Center  3  =  [6.26666667 2.64761905]
  Center  4  =  [5.14545455 2.39090909]
  Center  5  =  [4.76  3.184]
  Center  6  =  [5.77241379 2.84482759]
  Center  7  =  [7.475 3.125]
```



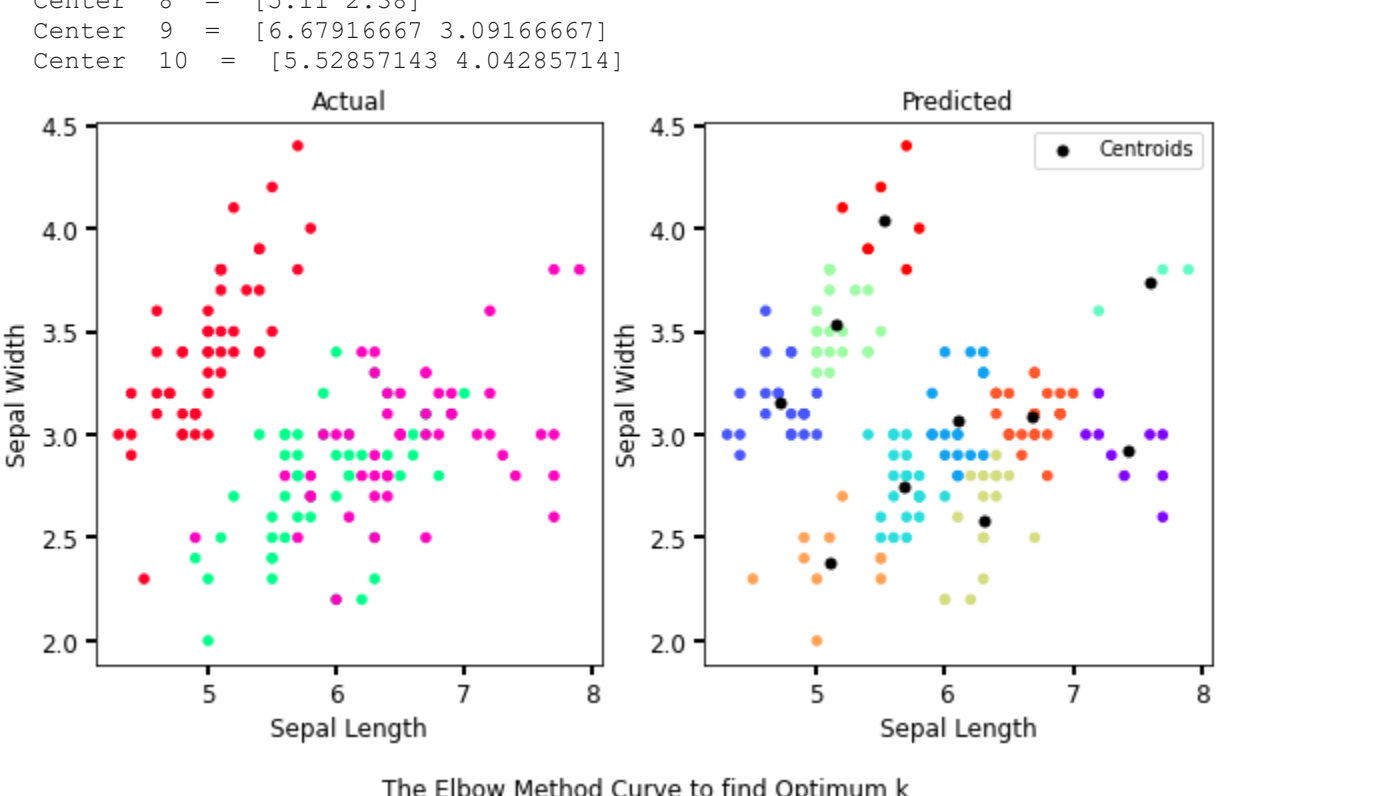
```
For k =  8

  Center  1  =  [5.29130435 3.7173913 ]
  Center  2  =  [6.26428571 3.02857143]
  Center  3  =  [7.625  3.0875]
  Center  4  =  [5.66071429 2.725     ]
  Center  5  =  [4.9        2.33333333]
  Center  6  =  [4.76  3.184]
  Center  7  =  [6.84761905 3.10952381]
  Center  8  =  [6.26 2.44]
```

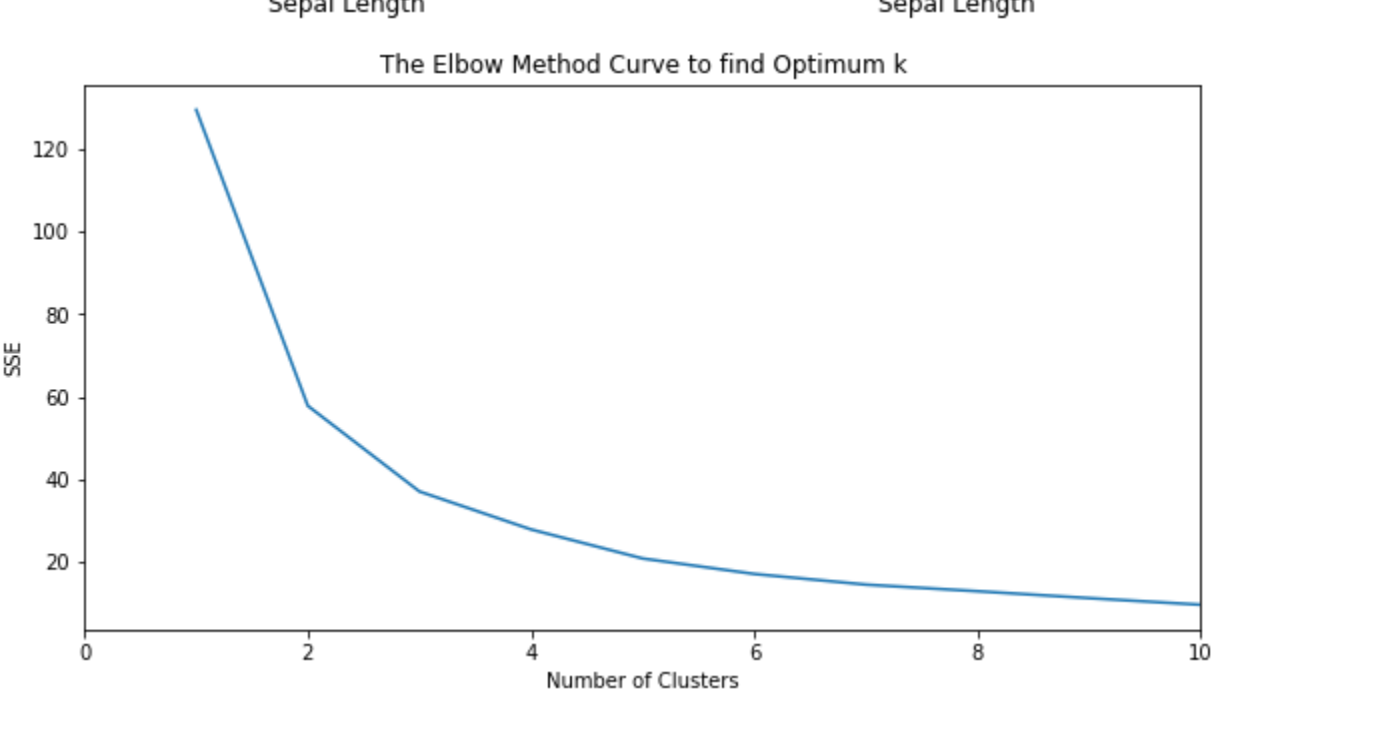


```
For k =  9

  Center  1  =  [5.64583333 2.70416667]
  Center  2  =  [6.73636364 3.07727273]
  Center  3  =  [5.29130435 3.7173913 ]
  Center  4  =  [4.94285714 2.38571429]
  Center  5  =  [4.76  3.184]
  Center  6  =  [7.475  2.9125]
  Center  7  =  [6.275   2.56875]
  Center  8  =  [6.16190476 3.07619048]
  Center  9  =  [7.6        3.73333333]
```



```
For k =  10

  Center  1  =  [7.43333333 2.92222222]
  Center  2  =  [4.70952381 3.15238095]
  Center  3  =  [6.10588235 3.07058824]
  Center  4  =  [5.68181818 2.75      ]
  Center  5  =  [7.6        3.73333333]
  Center  6  =  [5.155 3.53 ]
  Center  7  =  [6.3     2.58125]
  Center  8  =  [5.11 2.38]
  Center  9  =  [6.67916667 3.09166667]
  Center  10  =  [5.52857143 4.04285714]
```





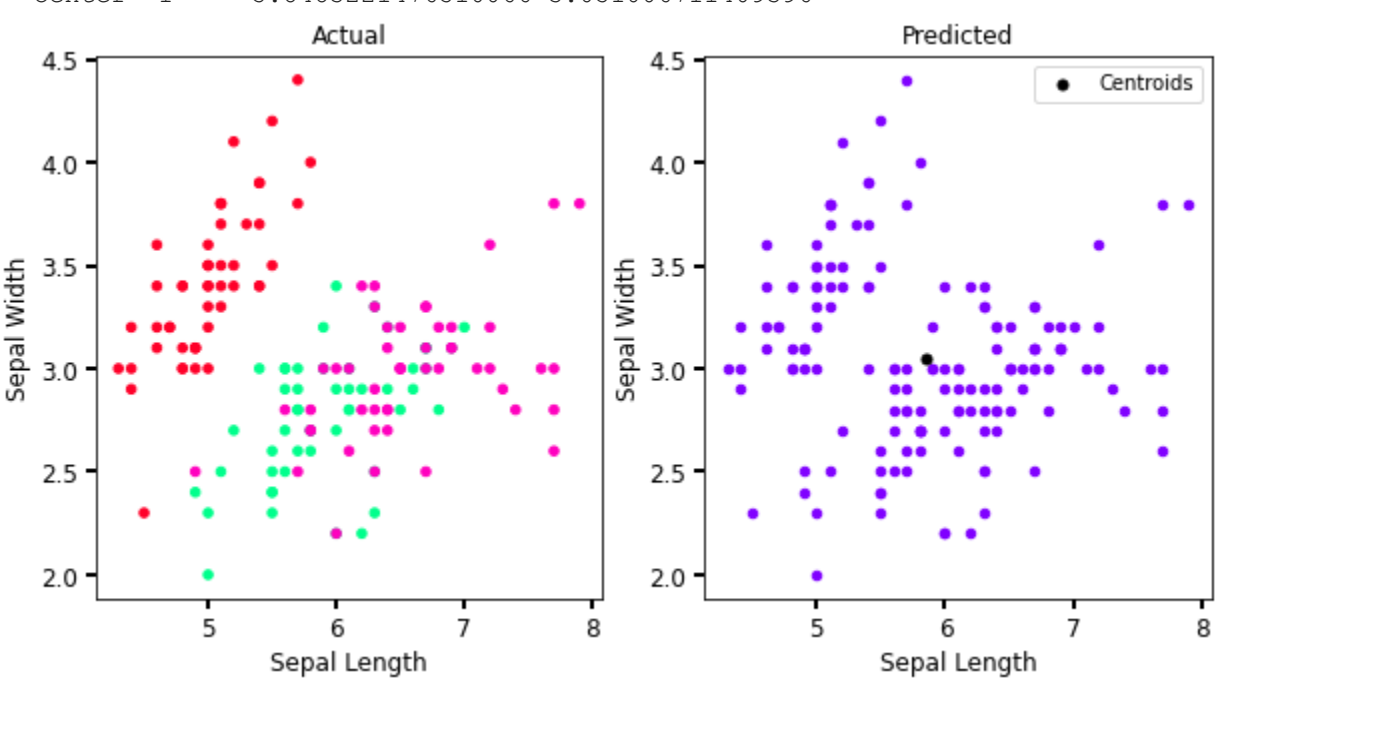
Exercise-8

Python Program for K Means Clustering (Manual).

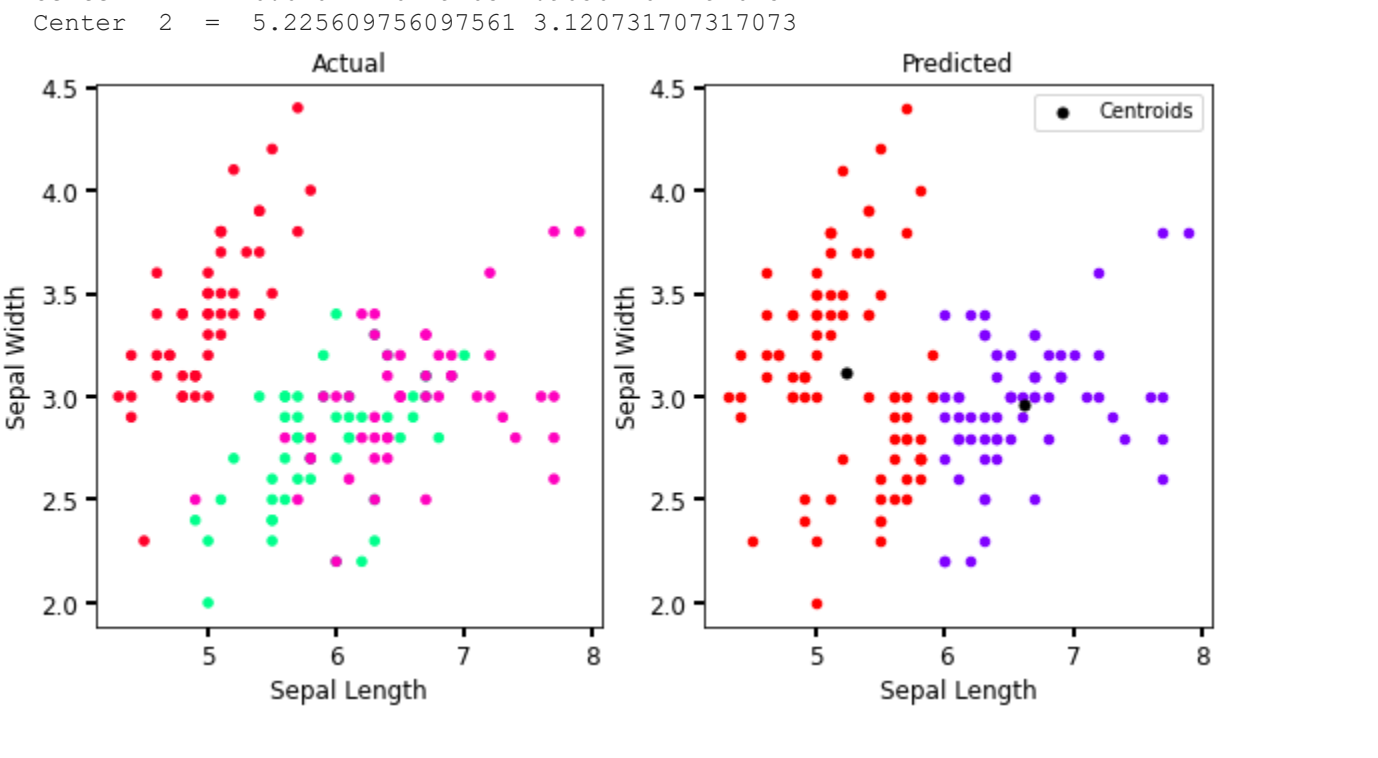
In [13]:

```python
# 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 of k
    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', 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(Centroids[0,:],Centroids[1,:],s=25,c='Black',label='Centroids')
    plt.legend()
    plt.show()

    # Calculating the sum of distances of samples to their closest cluster center
    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()
```

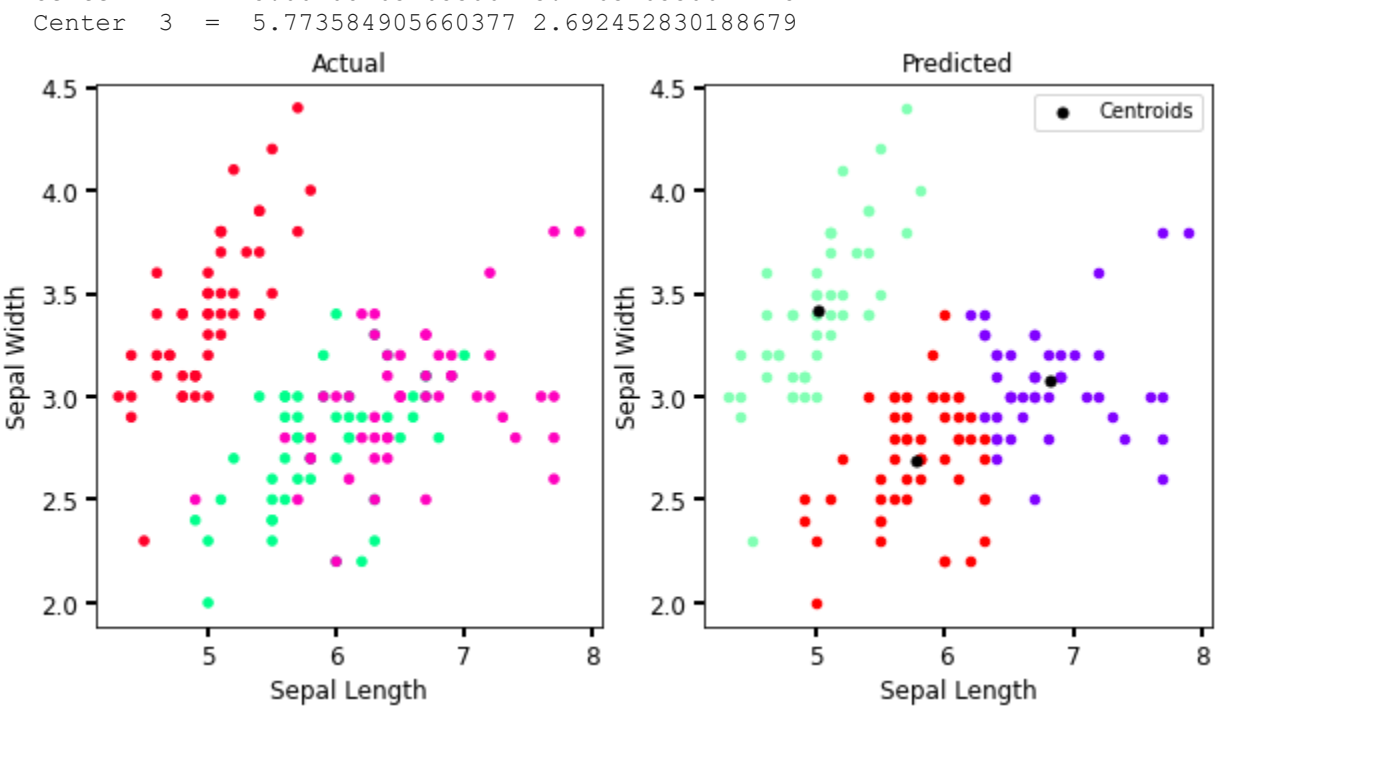
```
For K =  1
  Center  1  =  5.8483221476510066 3.051006711409396
```



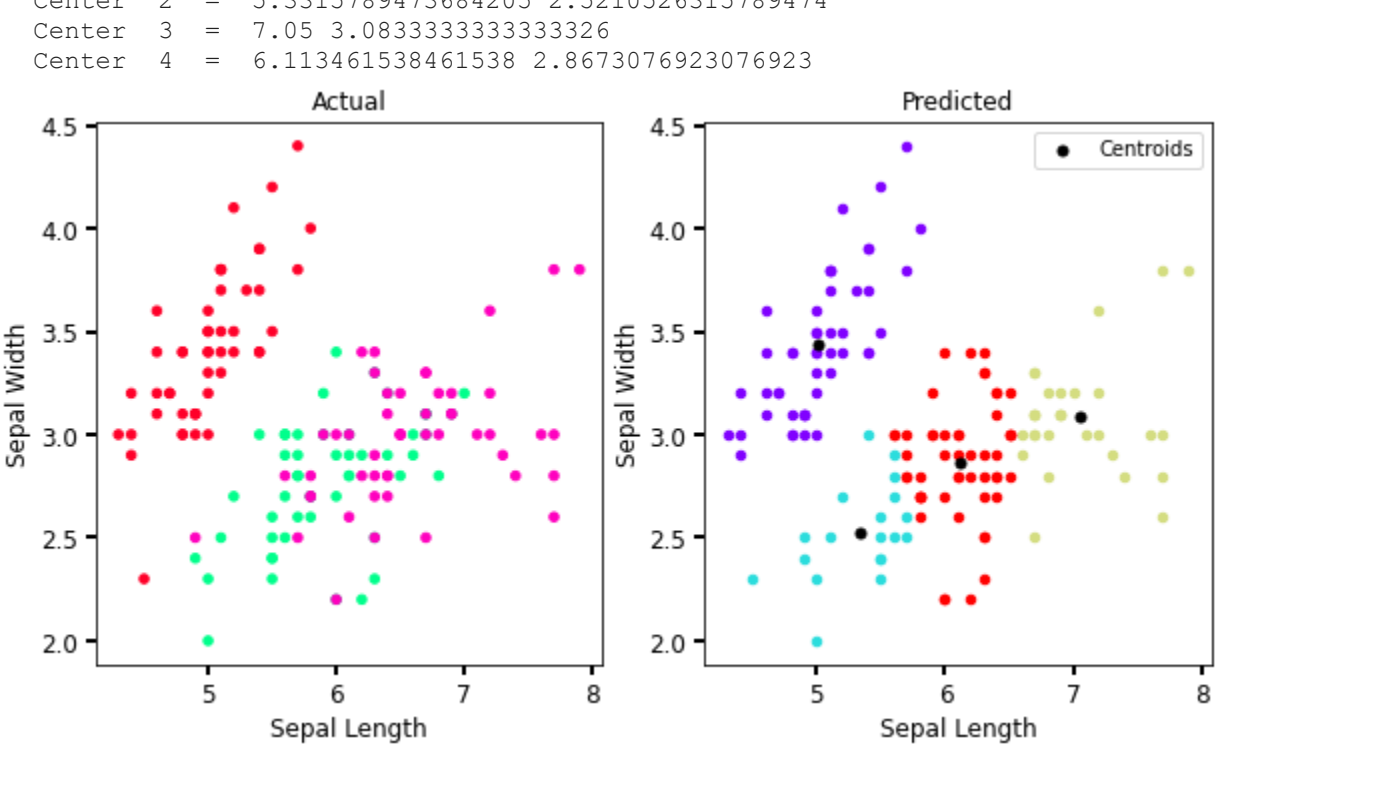
```
For K =  2
  Center  1  =  6.610447761194030 2.965671641791045
  Center  2  =  5.225609756097561 3.120731707317073
```



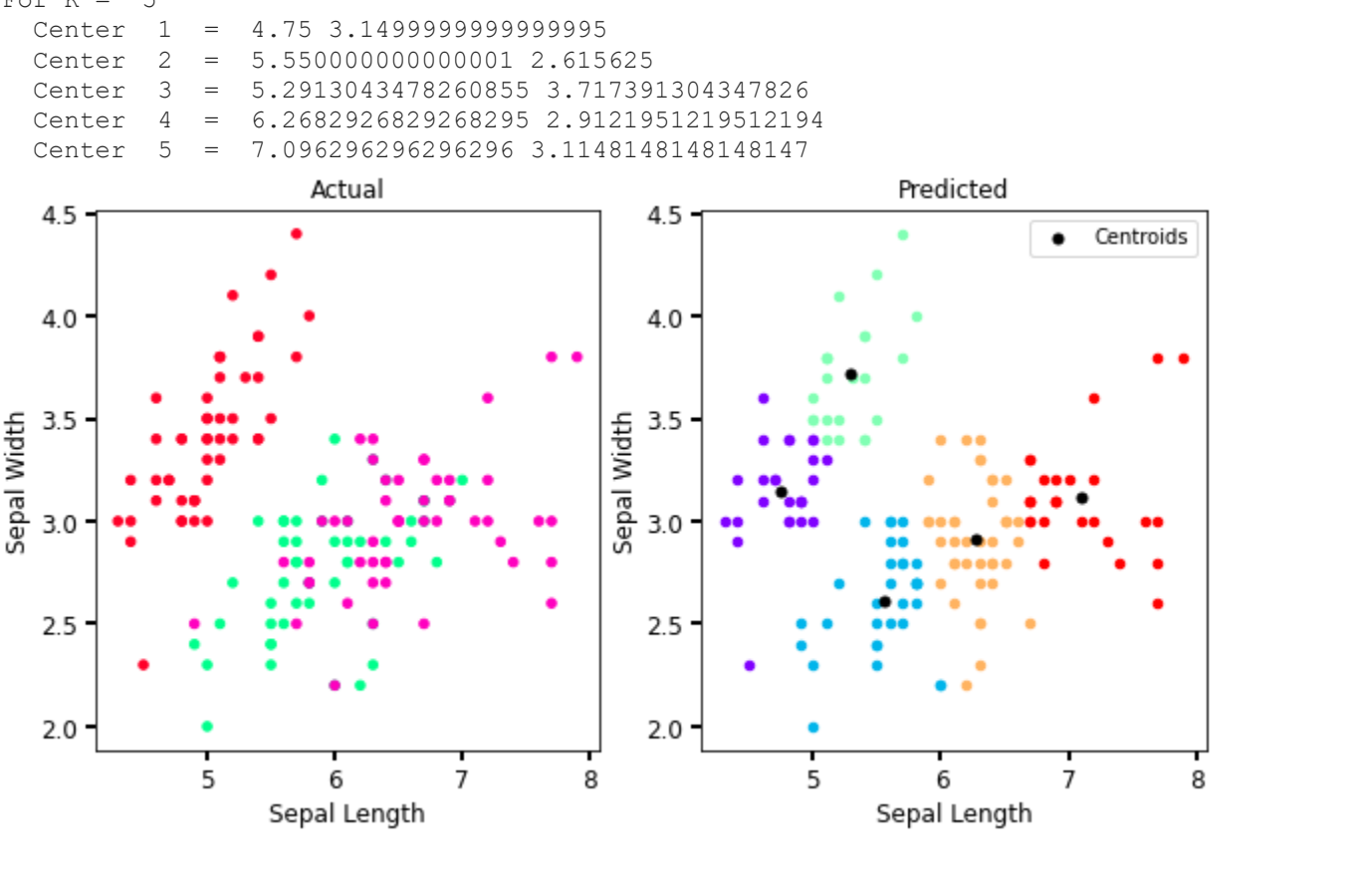
```
For K =  3
  Center  1  =  6.812765957446807 3.074468085106383
  Center  2  =  5.004081632653061 3.4163265306122454
  Center  3  =  5.773584905660377 2.692452830188679
```



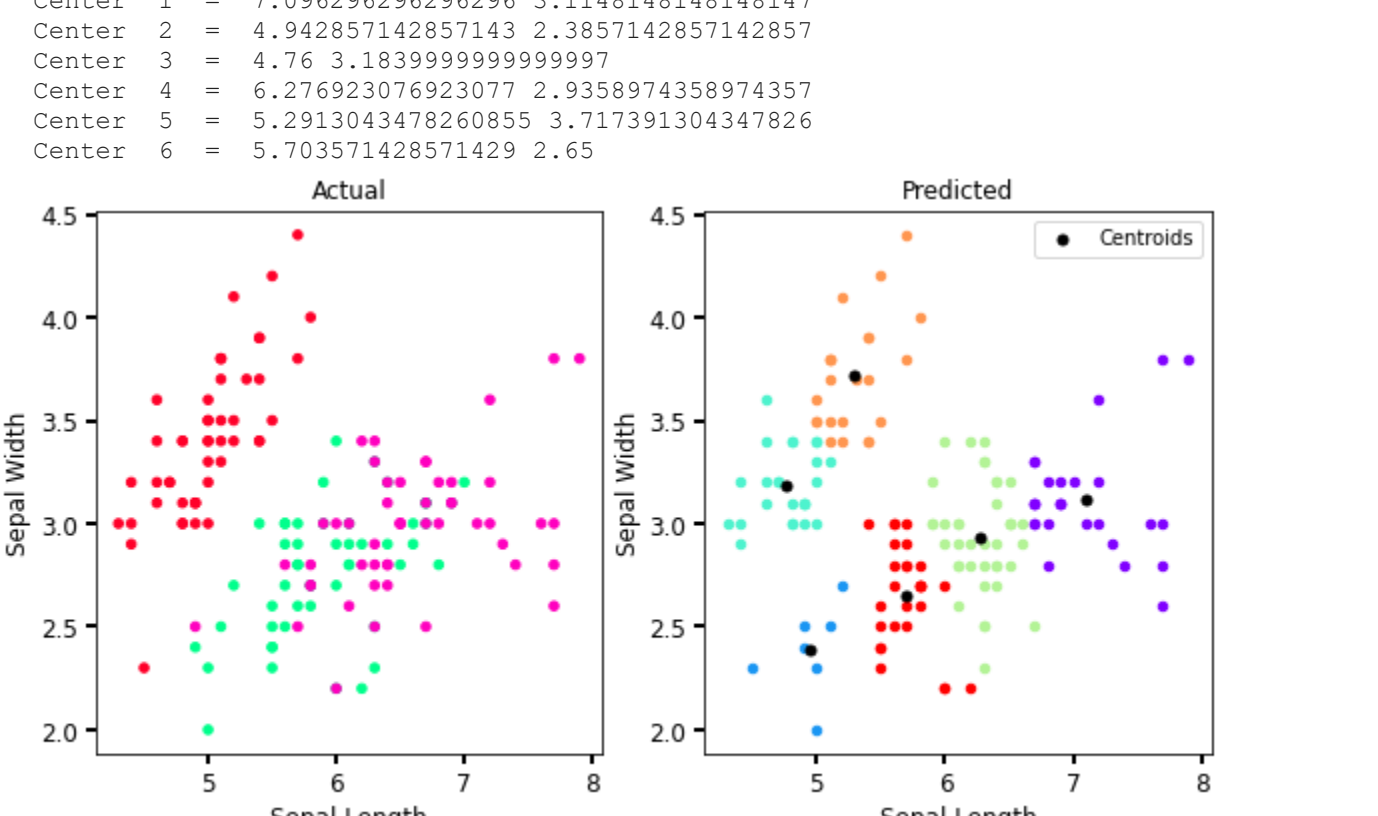
```
For K =  4
  Center  1  =  5.0145833333333333 3.4395833333333337
  Center  2  =  5.3315789473684205 2.5210526315789474
  Center  3  =  7.05 3.0833333333333326
  Center  4  =  6.113461538461538 2.8673076923076923
```



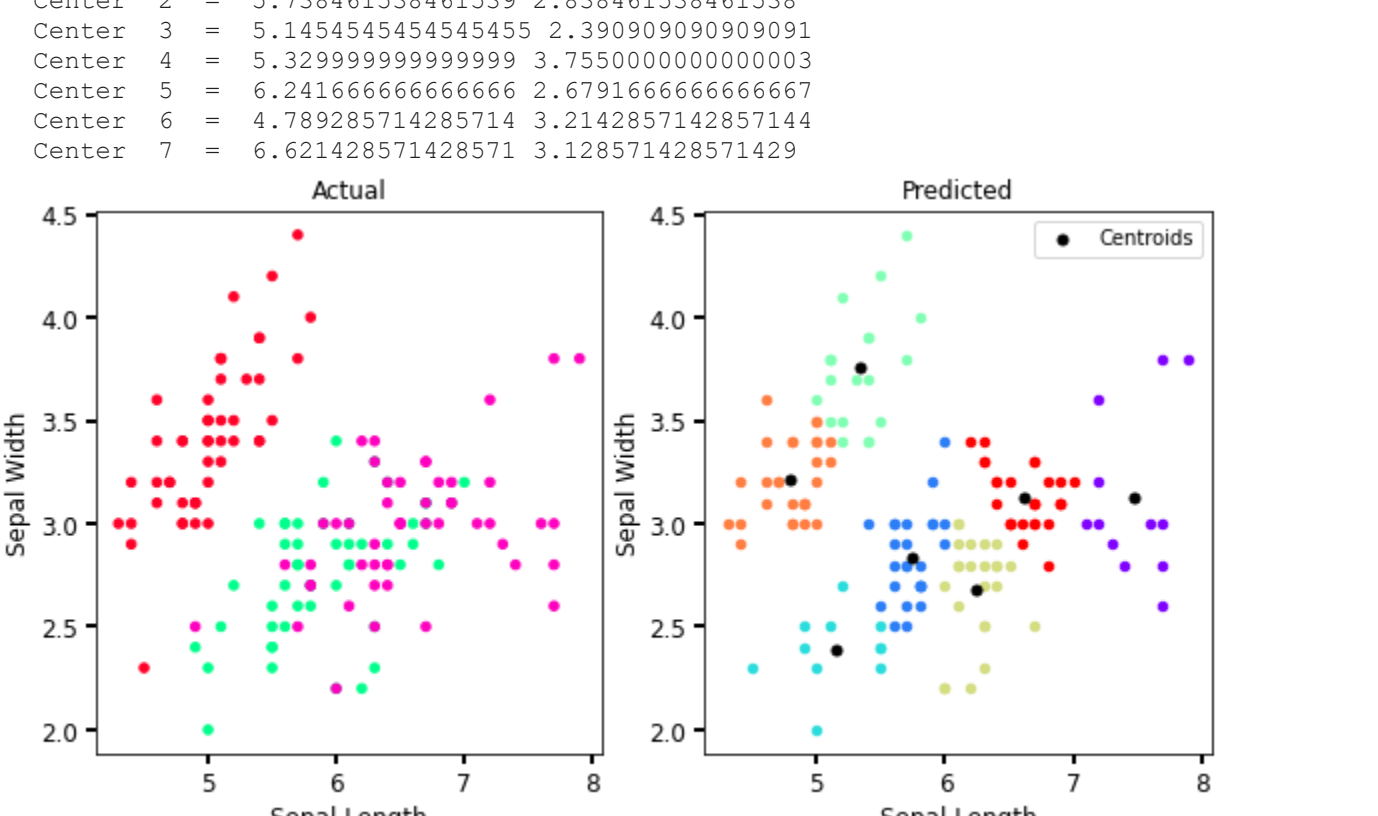
```
For K =  5
  Center  1  =  4.75 3.1499999999999995
  Center  2  =  5.550000000000001 2.615625
  Center  3  =  5.2913043478260855 3.717391304347826
  Center  4  =  6.2682926829268295 2.9121951219512194
  Center  5  =  7.096296296296296 3.1148148148148147
```



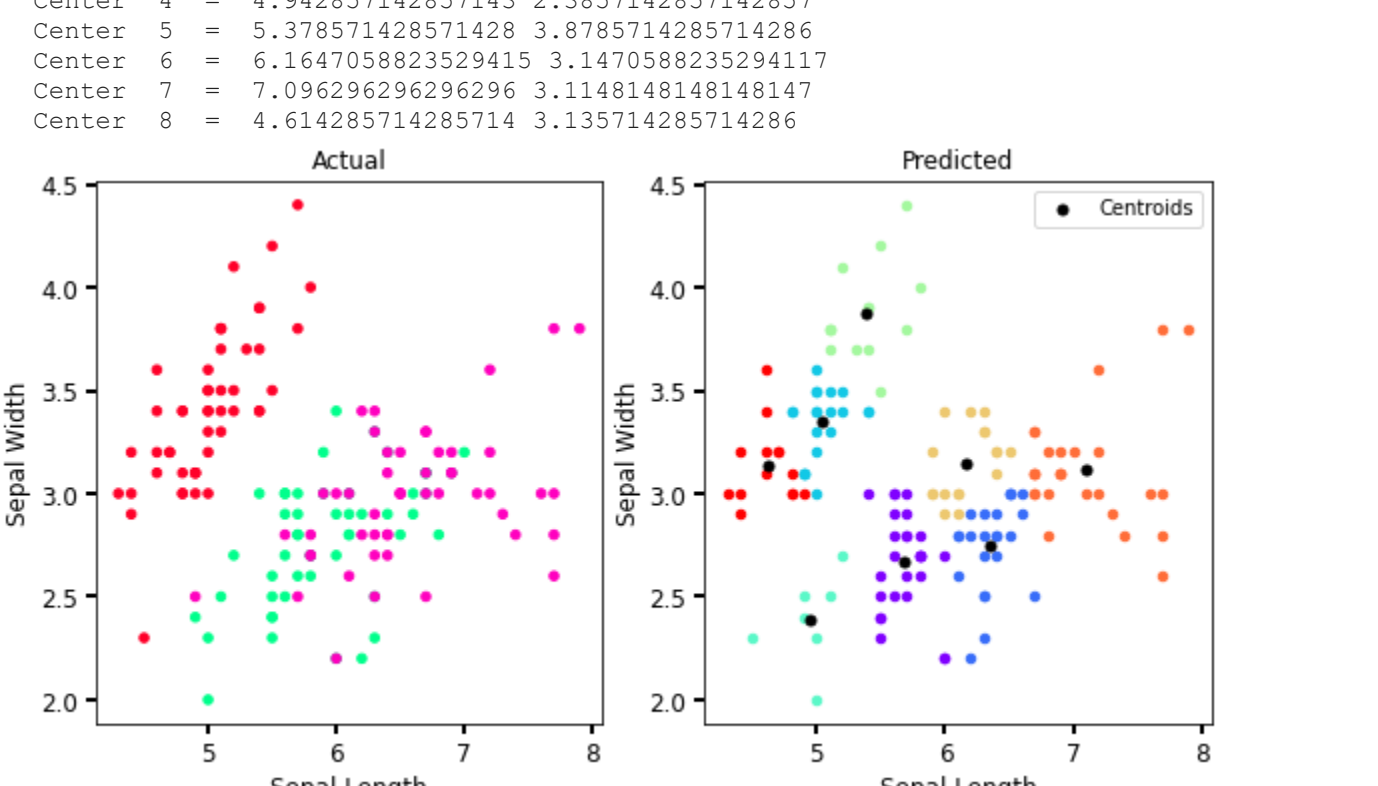
```
For K =  6
  Center  1  =  7.096296296296296 3.1148148148148147
  Center  2  =  4.942857142857143 2.3857142857142857
  Center  3  =  4.76 3.1839999999999997
  Center  4  =  6.276923076923077 2.9358974358974357
  Center  5  =  5.2913043478260855 3.717391304347826
  Center  6  =  5.703571428571429 2.65
```



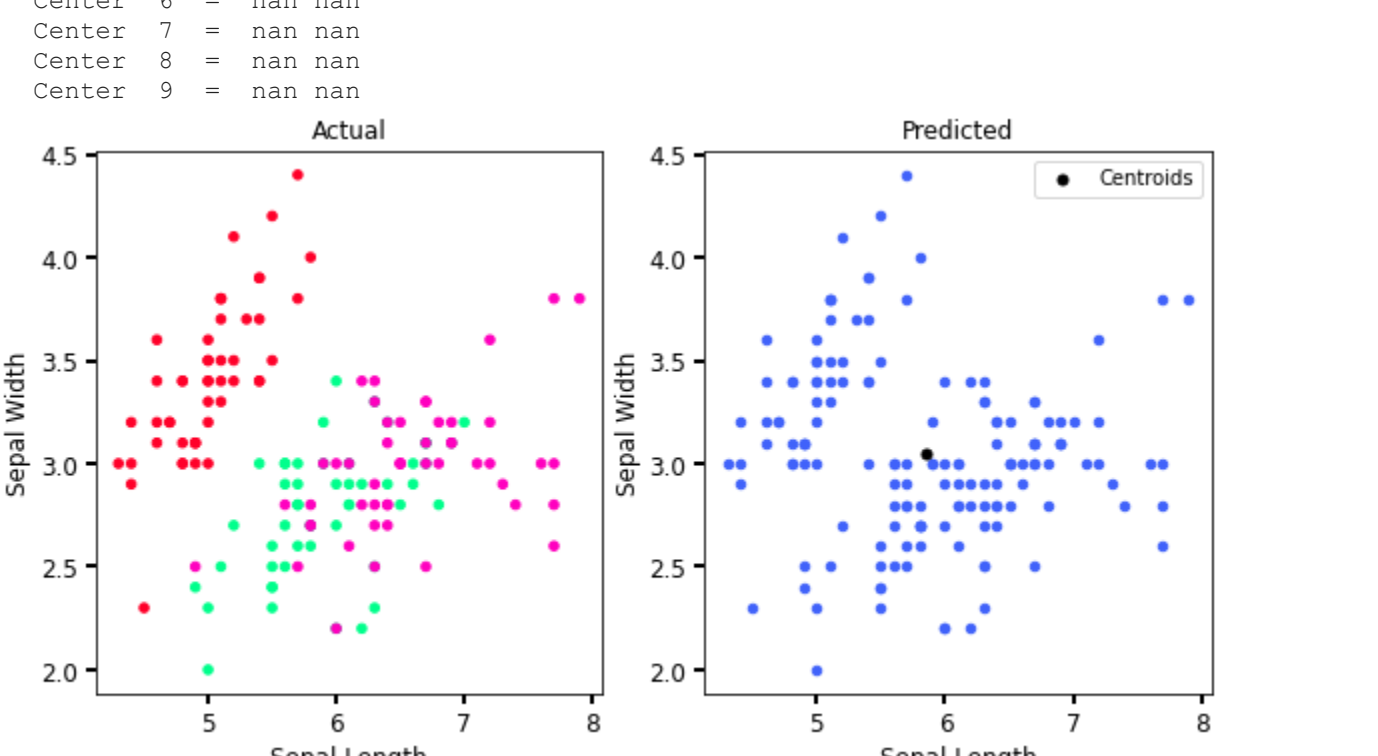
```
For K =  7
  Center  1  =  7.475000000000001 3.125
  Center  2  =  5.738461538461539 2.838461538461538
  Center  3  =  5.1454545454545455 2.390909090909091
  Center  4  =  5.329999999999999 3.7550000000000003
  Center  5  =  6.241666666666666 2.6791666666666667
  Center  6  =  4.789285714285714 3.2142857142857144
  Center  7  =  6.621428571428571 3.128571428571429
```



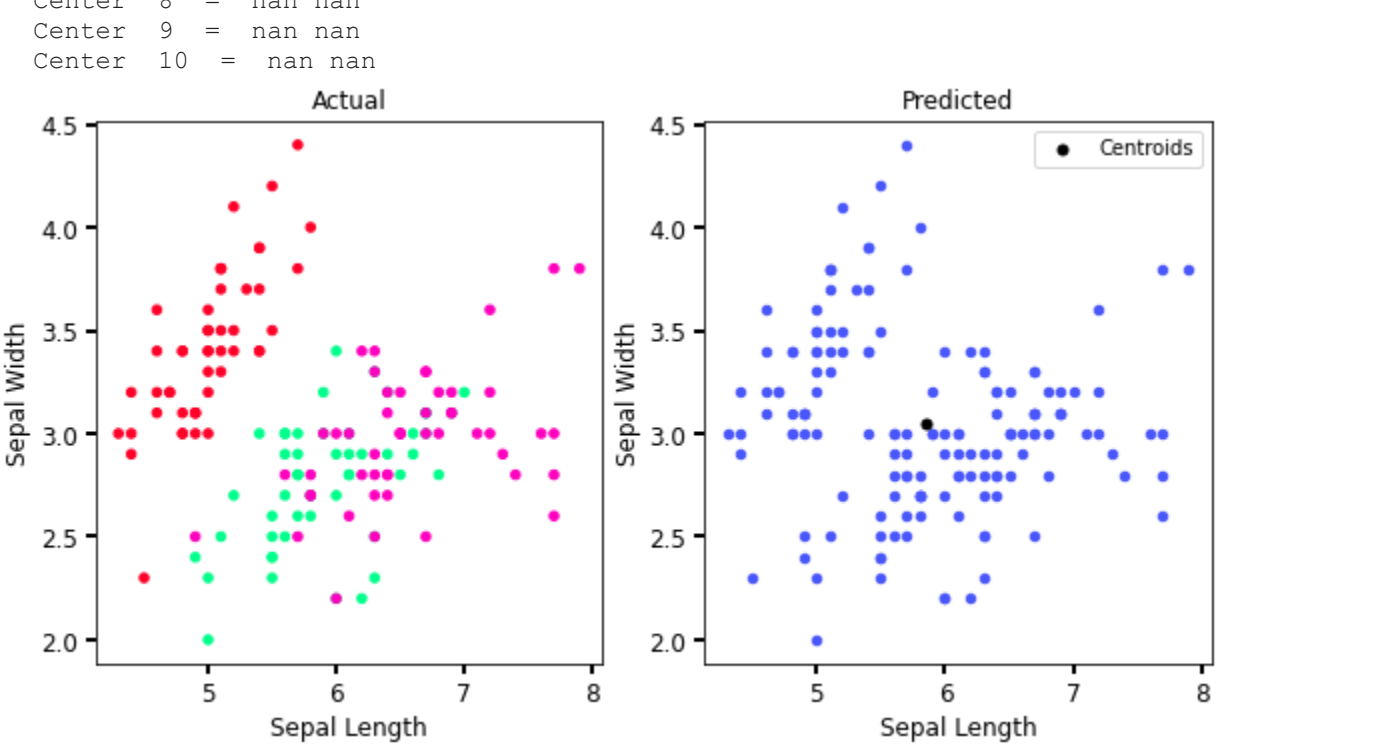
```
For K =  8
  Center  1  =  5.685185185185184 2.6666666666666665
  Center  2  =  6.356521739130436 2.7478260869565214
  Center  3  =  5.039999999999999 3.3449999999999998
  Center  4  =  4.942857142857143 2.3857142857142857
  Center  5  =  5.378571428571428 3.8785714285714286
  Center  6  =  6.164705882352941 3.1470588235294117
  Center  7  =  7.096296296296296 3.1148148148148147
  Center  8  =  4.614285714285714 3.135714285714286
```

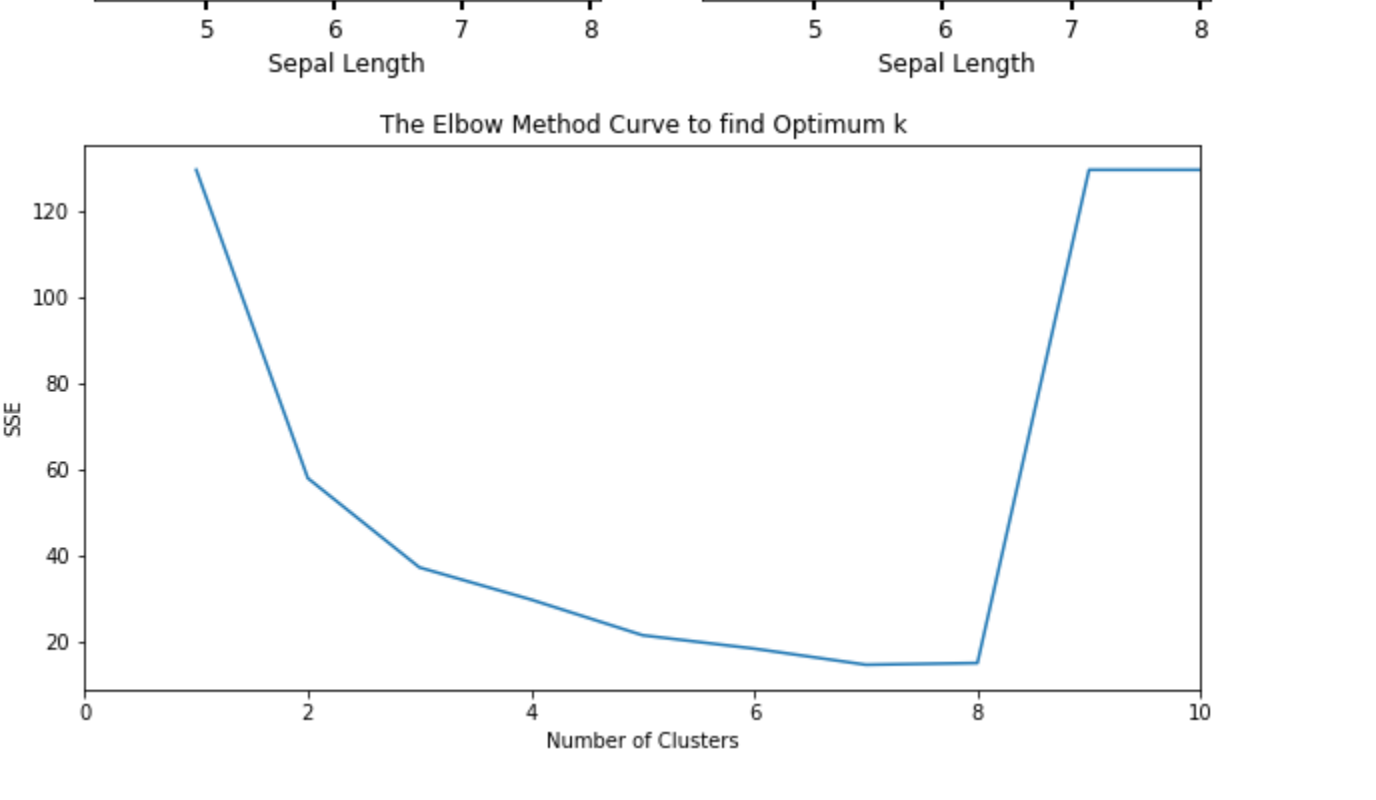


```
For K =  9
  Center  1  =  nan nan
  Center  2  =  5.8483221476510066 3.051006711409396
  Center  3  =  nan nan
  Center  4  =  nan nan
  Center  5  =  nan nan
  Center  6  =  nan nan
  Center  7  =  nan nan
  Center  8  =  nan nan
  Center  9  =  nan nan
```



```
For K =  10
  Center  1  =  nan nan
  Center  2  =  5.8483221476510066 3.051006711409396
  Center  3  =  nan nan
  Center  4  =  nan nan
  Center  5  =  nan nan
  Center  6  =  nan nan
  Center  7  =  nan nan
  Center  8  =  nan nan
  Center  9  =  nan nan
  Center  10  =  nan nan
```





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.