Foundations of Machine Learning (CS564)

Assignment No.1: K-Means & K-Medoids using Iris-Dataset

Output(s) File

Submitted by :

SAUMYEN MISHRA

2304RES06

Semester-I

M.Tech AI & DSE

IIT Patna

31st March'2024

1. **K-Means_Iris Python Code**

```
#*****************************************************************************************
******************************************************************************************
************************************************
#Importing CSV & Converting to dataframe
#*****************************************************************************************
******************************************************************************************
************************************************

import pandas as pd
import matplotlib.pyplot as plt
import math

url="https://raw.githubusercontent.com/SaumyenMishraIITP/iris/main/Iris.csv"    #    raw-
github-link
df=pd.read_csv(url)                                                             # converting
csv to dataframe
df=df.drop(['Id'], axis=1)                                                       # removing
the column 'Id'

#print (df)
#print(df.index)
#print(df.columns[3])

tot_rows=len(df.index)                                                           # Counting
no. of rows in dataframe
tot_cols=len(df.columns)                                                         # Counting
no. of columns in dataframe

#*****************************************************************************************
******************************************************************************************
************************************************
#Exploratory data analysis
#*****************************************************************************************
******************************************************************************************
************************************************

n_setosa=0
n_versicolor=0
n_virginica=0

sum_setosa_sep_len=0
sum_setosa_sep_wid=0
sum_setosa_pet_len=0
sum_setosa_pet_wid=0
```

```
sum_versicolor_sep_len=0
sum_versicolor_sep_wid=0
sum_versicolor_pet_len=0
sum_versicolor_pet_wid=0

sum_virginica_sep_len=0
sum_virginica_sep_wid=0
sum_virginica_pet_len=0
sum_virginica_pet_wid=0

for i in range(0,tot_rows,1) :
  if  (df.iat[i,4]=="Iris-setosa") :
    df.iat[i,4] = 0
    n_setosa = n_setosa+1

    sum_setosa_sep_len = sum_setosa_sep_len + df.iat[i,0]
    sum_setosa_sep_wid= sum_setosa_sep_wid + df.iat[i,1]
    sum_setosa_pet_len = sum_setosa_pet_len + df.iat[i,2]
    sum_setosa_pet_wid = sum_setosa_pet_wid + df.iat[i,3]

  elif  (df.iat[i,4]=="Iris-versicolor") :
    df.iat[i,4] = 1
    n_versicolor = n_versicolor+1

    sum_versicolor_sep_len = sum_versicolor_sep_len + df.iat[i,0]
    sum_versicolor_sep_wid= sum_versicolor_sep_wid + df.iat[i,1]
    sum_versicolor_pet_len = sum_versicolor_pet_len + df.iat[i,2]
    sum_versicolor_pet_wid = sum_versicolor_pet_wid + df.iat[i,3]

  elif (df.iat[i,4]=="Iris-virginica") :
      df.iat[i,4] = 2
      n_virginica = n_virginica+1

      sum_virginica_sep_len = sum_virginica_sep_len + df.iat[i,0]
      sum_virginica_sep_wid = sum_virginica_sep_wid + df.iat[i,1]
      sum_virginica_pet_len = sum_virginica_pet_len + df.iat[i,2]
      sum_virginica_pet_wid = sum_virginica_pet_wid + df.iat[i,3]


mean_setosa_sep_len = sum_setosa_sep_len/n_setosa
mean_setosa_sep_wid = sum_setosa_sep_wid/n_setosa
mean_setosa_pet_len = sum_setosa_pet_len/n_setosa
mean_setosa_pet_wid = sum_setosa_pet_wid/n_setosa

mean_versicolor_sep_len = sum_versicolor_sep_len/n_versicolor
mean_versicolor_sep_wid = sum_versicolor_sep_wid/n_versicolor
mean_versicolor_pet_len = sum_versicolor_pet_len/n_versicolor
mean_versicolor_pet_wid = sum_versicolor_pet_wid/n_versicolor

mean_virginica_sep_len = sum_virginica_sep_len/n_virginica
mean_virginica_sep_wid = sum_virginica_sep_wid/n_virginica
```

```
mean_virginica_pet_len = sum_virginica_pet_len/n_virginica
mean_virginica_pet_wid = sum_virginica_pet_wid/n_virginica

initial_centroids=[[mean_setosa_sep_len,mean_setosa_sep_wid,mean_setosa_pet_len,mean_seto
sa_pet_wid],
[mean_versicolor_sep_len,mean_versicolor_sep_wid,mean_versicolor_pet_len,mean_versicolor_
pet_wid],
[mean_virginica_sep_len,mean_virginica_sep_wid,mean_virginica_pet_len,mean_virginica_pet_
wid]]


#print(f"\n{mean_setosa_sep_len}\n{mean_setosa_sep_wid}\n{mean_setosa_pet_len}\n{mean_set
osa_pet_wid}")
#print(f"\n{mean_versicolor_sep_len}\n{mean_versicolor_sep_wid}\n{mean_versicolor_pet_len
}\n{mean_versicolor_pet_wid}")
#print(f"\n{mean_virginica_sep_len}\n{mean_virginica_sep_wid}\n{mean_virginica_pet_len}\n
{mean_virginica_pet_wid}")

#print (df)

#****************************************************************************************
*****************************************************************************************
***********************************************
#SCATTER PLOT OF ORIGINAL DATASET
#****************************************************************************************
*****************************************************************************************
***********************************************
sepallen=[]
sepalwid=[]
petallen=[]
petalwid=[]
sepallen_X_petallen=[]
sepalwid_X_petalwid=[]

for p in range(0, len(df.index), 1) :
  sepallen.append(df.iat[p,0])
  sepalwid.append(df.iat[p,1])
  petallen.append(df.iat[p,2])
  petalwid.append(df.iat[p,3])
  sepallen_X_petallen.append(((7*df.iat[p,0]) + (3*df.iat[p,2])))
  sepalwid_X_petalwid.append(((11*(df.iat[p,1])) + (df.iat[p,3])))

plt.scatter(sepallen,sepalwid, c='green')
plt.xlabel('sepal length in cm')
plt.ylabel('sepal width in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(petallen,petalwid,c='magenta')
plt.xlabel('Petal length in cm')
plt.ylabel('Petal width in cm')
```

```
plt.title('Raw-data')
plt.show()

plt.scatter(sepallen,petallen,c='cyan')
plt.xlabel('Sepal length in cm')
plt.ylabel('Petal length in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepallen,petalwid,c='yellow')
plt.xlabel('Sepal length in cm')
plt.ylabel('Petal width in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepalwid,petalwid,c='blue')
plt.xlabel('Sepal width in cm')
plt.ylabel('Petal width in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepalwid,petallen,c='black')
plt.xlabel('Sepal width in cm')
plt.ylabel('Petal length in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepallen_X_petallen,sepalwid_X_petalwid,c='red')
plt.xlabel('Linear Sum of Sepal & Petal lengths in cm')
plt.ylabel('Linear Sum  of Sepal & Petal widths in cm')
plt.title('Raw-data')
plt.show()

#***************************************************************************************
***************************************************************************************
**********************************************
#Centroid Function, SSE calculation Function, K-Means Assignment function, K-Means-Iris
Function, Shannon's Entropy, Scatter Plot of Clusterings & Handling Empty Clusters
#***************************************************************************************
***************************************************************************************
**********************************************

#old_centroids_with_cluster=[[0,mean_setosa_sep_len,mean_setosa_sep_wid,mean_setosa_pet_l
en,mean_setosa_pet_wid],
[1,mean_versicolor_sep_len,mean_versicolor_sep_wid,mean_versicolor_pet_len,mean_versicolo
r_pet_wid],
[2,mean_virginica_sep_len,mean_virginica_sep_wid,mean_virginica_pet_len,mean_virginica_pe
t_wid]]

def centroid_calc_function (cluster) :
  R=len(cluster.index)
```

```
  C=len(cluster.columns)

  sum1=[]
  centroid_coordinates=[]

  for i in range (0,C,1) :
    sum1.append(0)
    centroid_coordinates.append(0)

  for j in range (0,R,1) :
    for k in range(0,C,1) :
      sum1[k] = sum1[k] + cluster.iat[j,k]

  for m in range (0,C,1) :
    if (R!=0) :
      centroid_coordinates[m]=sum1[m]/R
    else :
      break

  print("\nEmpty Clusters do not have a defined centroid")

  #print (centroid_coordinates)

  return centroid_coordinates

#***************************************************************************************
*************************************************

#***************************************************************************************
*************************************************

def SSE(cluster) :

  rows_1 = len(cluster.index)

  if (rows_1 == 0) :
    print("Empty Clusters do not have a defined SSE")

  else :
    cen = centroid_calc_function(cluster)
    SSE_Cluster=0

    for i in range (0,rows_1,1) :
          SSE_Cluster = SSE_Cluster + (((cen[0]-cluster.iat[i,0])**2) + ((cen[1]-
cluster.iat[i,1])**2) + ((cen[2]-cluster.iat[i,2])**2) + ((cen[3]-cluster.iat[i,3])**2))

  return SSE_Cluster
```

```
#****************************************************************************************
*****************************************************************************************
**************************************************
#****************************************************************************************
*****************************************************************************************
**************************************************

def assignment_function(centroids,dataset) :

  num_rows = len(dataset.index)
  num_cols = len(dataset.columns)

  k1=0
  k2=0
  k3=0

  dist1=[]
  dist2=[]
  dist3=[]

  for i in range(0,num_rows,1) :
      k1=math.sqrt(((centroids[0][0] - dataset.iat[i,0])**2) + ((centroids[0][1] - dataset.iat[i,1])**2) + ((centroids[0][2] - dataset.iat[i,2])**2) + ((centroids[0][3] - dataset.iat[i,3])**2))
    dist1.append(k1)

      k2=math.sqrt(((centroids[1][0] - dataset.iat[i,0])**2) + ((centroids[1][1] - dataset.iat[i,1])**2) + ((centroids[1][2] - dataset.iat[i,2])**2) + ((centroids[1][3] - dataset.iat[i,3])**2))
    dist2.append(k2)

      k3=math.sqrt(((centroids[2][0] - dataset.iat[i,0])**2) + ((centroids[2][1] - dataset.iat[i,1])**2) + ((centroids[2][2] - dataset.iat[i,2])**2) + ((centroids[2][3] - dataset.iat[i,3])**2))
    dist3.append(k3)

  list0=[]
  list1=[]
  list2=[]

  for j in range (0,num_rows,1) :
    if (min(dist1[j],dist2[j],dist3[j])==dist1[j]) :
      a0_0=dataset.iat[j,0]
      a0_1=dataset.iat[j,1]
      a0_2=dataset.iat[j,2]
      a0_3=dataset.iat[j,3]
      a0=[a0_0, a0_1, a0_2, a0_3]
      list0.append(a0)

    elif (min(dist1[j],dist2[j],dist3[j])==dist2[j]) :
      a1_0=dataset.iat[j,0]
```

```
      a1_1=dataset.iat[j,1]
      a1_2=dataset.iat[j,2]
      a1_3=dataset.iat[j,3]
      a1=[a1_0, a1_1, a1_2, a1_3]
      list1.append(a1)

    elif (min(dist1[j],dist2[j],dist3[j])==dist3[j]) :
      a2_0=dataset.iat[j,0]
      a2_1=dataset.iat[j,1]
      a2_2=dataset.iat[j,2]
      a2_3=dataset.iat[j,3]
      a2=[a2_0, a2_1, a2_2, a2_3]
      list2.append(a2)

              cluster0              =              pd.DataFrame            (list(list0),
columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"])

              cluster1              =              pd.DataFrame            (list(list1),
columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"])

              cluster2              =              pd.DataFrame            (list(list2),
columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"])

  centroid_cluster0 = centroid_calc_function(cluster0)
  centroid_cluster1 = centroid_calc_function(cluster1)
  centroid_cluster2 = centroid_calc_function(cluster2)

  new_iteration_centroids = [centroid_cluster0,centroid_cluster1,centroid_cluster2]

  #print(new_iteration_centroids)
  #print (f"\nIris-Setosa cluster looks like :\n\n{cluster0}\n\n")
  #print (f"\nIris-Versicolor cluster looks like :\n\n{cluster1}\n\n")
  #print (f"\nIris-Virginica cluster looks like :\n\n{cluster2}\n\n")
  #cluster0, cluster1, cluster2, new_iteration_centroids

  return cluster0, cluster1, cluster2, new_iteration_centroids

#*****************************************************************************************
*****************************************************************************************
**************************************************
#*****************************************************************************************
*****************************************************************************************
**************************************************

def kmeans_iris(ini_centroids,dataset) :

  K=3
  Last_iteration_centroids = ini_centroids
  Iterations=0

  while 1:
```

```
    Iterations = Iterations+1

                   [C_0,         C_1,         C_2,         New_iteration_centroids]         =
assignment_function(Last_iteration_centroids,dataset)
    len_C_0 = len(C_0.index)
    len_C_1 = len(C_1.index)
    len_C_2 = len(C_2.index)

    #and len_C_0!=0 and len_C_1!=0 and len_C_2!=0

     if(New_iteration_centroids==Last_iteration_centroids  and  len_C_0!=0  and  len_C_1!=0
and len_C_2!=0) :

print("\n\n******************************************************************************
*****************************************************************************************
***************************")

print("\n********************************************************************************
*****************************************************************************************
*************************")
      print("\nCASE: 1 COMMON\n")

       print(f"\n\nConvergence  Achieved  in  th  K-Means  Clustering  Algorithm!!!\n\nTotal
Number of Iterations : {Iterations}\n\nTotal Number of Clusters : {K}\n\nTotal Number of
Points  in  First  Cluster  :  {len(C_0.index)}\n\nSSE  for  the  First  Cluster  :  {SSE(C_0)}
cm.Sq.\n\nTotal  Number  of  Points  in  Second  Cluster  :  {len(C_1.index)}\n\nSSE  for  the
Second   Cluster   :   {SSE(C_1)}   cm.Sq.\n\nTotal   Number   of   Points   in   Third   Cluster   :
{len(C_2.index)}\n\nSSE for the Third Cluster : {SSE(C_2)} cm.Sq.\n\nTotal SSE for the
finalized 3-Cluster-System : {SSE(C_0) + SSE(C_1) + SSE(C_2)} cm.Sq.\n\n")

      sepallen_C_0=[]
      sepallen_C_1=[]
      sepallen_C_2=[]

      sepalwid_C_0=[]
      sepalwid_C_1=[]
      sepalwid_C_2=[]

      petallen_C_0=[]
      petallen_C_1=[]
      petallen_C_2=[]

      petalwid_C_0=[]
      petalwid_C_1=[]
      petalwid_C_2=[]

      sepallen_X_petallen_C_0=[]
      sepallen_X_petallen_C_1=[]
      sepallen_X_petallen_C_2=[]
```

```python
sepalwid_X_petalwid_C_0=[]
sepalwid_X_petalwid_C_1=[]
sepalwid_X_petalwid_C_2=[]

for z in range (0,len(C_0.index),1) :
  sepallen_C_0.append(C_0.iat[z,0])
  sepalwid_C_0.append(C_0.iat[z,1])
  petallen_C_0.append(C_0.iat[z,2])
  petalwid_C_0.append(C_0.iat[z,3])
  sepallen_X_petallen_C_0.append(((7*C_0.iat[z,0])) + (3*C_0.iat[z,2]))
  sepalwid_X_petalwid_C_0.append((11*C_0.iat[z,1]) + (C_0.iat[z,3]))

for y in range (0,len(C_1.index),1) :
  sepallen_C_1.append(C_1.iat[y,0])
  sepalwid_C_1.append(C_1.iat[y,1])
  petallen_C_1.append(C_1.iat[y,2])
  petalwid_C_1.append(C_1.iat[y,3])
  sepallen_X_petallen_C_1.append(((7*C_1.iat[y,0])) + (3*C_1.iat[y,2]))
  sepalwid_X_petalwid_C_1.append((11*C_1.iat[y,1]) + (C_1.iat[y,3]))

for x in range (0,len(C_2.index),1) :
  sepallen_C_2.append(C_2.iat[x,0])
  sepalwid_C_2.append(C_2.iat[x,1])
  petallen_C_2.append(C_2.iat[x,2])
  petalwid_C_2.append(C_2.iat[x,3])
  sepallen_X_petallen_C_2.append(((7*C_2.iat[x,0])) + (3*C_2.iat[x,2]))
  sepalwid_X_petalwid_C_2.append((11*C_2.iat[x,1]) + (C_2.iat[x,3]))

plt.scatter(sepallen_C_0,sepalwid_C_0, c='green')
plt.scatter(sepallen_C_1,sepalwid_C_1, c='blue')
plt.scatter(sepallen_C_2,sepalwid_C_2, c='red')
plt.xlabel('sepal length in cm')
plt.ylabel('sepal width in cm')
plt.title('Clusterings')
plt.show()

plt.scatter(petallen_C_0,petalwid_C_0, c='green')
plt.scatter(petallen_C_1,petalwid_C_1, c='blue')
plt.scatter(petallen_C_2,petalwid_C_2, c='red')
plt.xlabel('petal length in cm')
plt.ylabel('petal width in cm')
plt.title('Clusterings')
plt.show()

plt.scatter(sepallen_C_0,petallen_C_0, c='green')
plt.scatter(sepallen_C_1,petallen_C_1, c='blue')
plt.scatter(sepallen_C_2,petallen_C_2, c='red')
plt.xlabel('sepal length in cm')
plt.ylabel('petal length in cm')
plt.title('Clusterings')
plt.show()
```

```
    plt.scatter(sepallen_C_0,petalwid_C_0, c='green')
    plt.scatter(sepallen_C_1,petalwid_C_1, c='blue')
    plt.scatter(sepallen_C_2,petalwid_C_2, c='red')
    plt.xlabel('sepal length in cm')
    plt.ylabel('petal width in cm')
    plt.title('Clusterings')
    plt.show()

    plt.scatter(sepalwid_C_0,petalwid_C_0, c='green')
    plt.scatter(sepalwid_C_1,petalwid_C_1, c='blue')
    plt.scatter(sepalwid_C_2,petalwid_C_2, c='red')
    plt.xlabel('sepal width in cm')
    plt.ylabel('petal width in cm')
    plt.title('Clusterings')
    plt.show()

    plt.scatter(sepallen_X_petallen_C_0,sepalwid_X_petalwid_C_0, c='green')
    plt.scatter(sepallen_X_petallen_C_1,sepalwid_X_petalwid_C_1, c='blue')
    plt.scatter(sepallen_X_petallen_C_2,sepalwid_X_petalwid_C_2, c='red')
    plt.xlabel('Linear Sum  of Sepal & Petal lengths in cm')
    plt.ylabel('Linear Sum  of Sepal & Petal widths in cm')
    plt.title('Clusterings')
    plt.show()

    break

  elif (New_iteration_centroids!=Last_iteration_centroids and len_C_0!=0 and len_C_1!=0 
and len_C_2!=0) :
    print("\nCASE: 2 COMMON\n")
    Last_iteration_centroids = New_iteration_centroids

  elif (len_C_0==0 and len_C_1==0) :
    print("\nCASE: 3 VERY RARE\n")
    Last_iteration_centroids[2]=New_iteration_centroids[2]

    Last_iteration_centroids[0][0]=0.5*New_iteration_centroids[2][0]
    Last_iteration_centroids[0][1]=0.5*New_iteration_centroids[2][1]
    Last_iteration_centroids[0][2]=0.5*New_iteration_centroids[2][2]
    Last_iteration_centroids[0][3]=0.5*New_iteration_centroids[2][3]

    Last_iteration_centroids[1][0]=0.75*New_iteration_centroids[2][0]
    Last_iteration_centroids[1][1]=0.75*New_iteration_centroids[2][1]
    Last_iteration_centroids[1][2]=0.75*New_iteration_centroids[2][2]
    Last_iteration_centroids[1][3]=0.75*New_iteration_centroids[2][3]

  elif (len_C_0==0 and len_C_2==0) :
    print("\nCASE: 4 VERY RARE\n")
    Last_iteration_centroids[1]=New_iteration_centroids[1]

    Last_iteration_centroids[0][0]=0.5*New_iteration_centroids[1][0]
```

```
      Last_iteration_centroids[0][1]=0.5*New_iteration_centroids[1][1]
      Last_iteration_centroids[0][2]=0.5*New_iteration_centroids[1][2]
      Last_iteration_centroids[0][3]=0.5*New_iteration_centroids[1][3]

      Last_iteration_centroids[2][0]=0.75*New_iteration_centroids[1][0]
      Last_iteration_centroids[2][1]=0.75*New_iteration_centroids[1][1]
      Last_iteration_centroids[2][2]=0.75*New_iteration_centroids[1][2]
      Last_iteration_centroids[2][3]=0.75*New_iteration_centroids[1][3]

    elif (len_C_1==0 and len_C_2==0) :
      print("\nCASE: 5 VERY RARE\n")
      Last_iteration_centroids[0]=New_iteration_centroids[0]

      Last_iteration_centroids[1][0]=0.5*New_iteration_centroids[0][0]
      Last_iteration_centroids[1][1]=0.5*New_iteration_centroids[0][1]
      Last_iteration_centroids[1][2]=0.5*New_iteration_centroids[0][2]
      Last_iteration_centroids[1][3]=0.5*New_iteration_centroids[0][3]

      Last_iteration_centroids[2][0]=0.75*New_iteration_centroids[0][0]
      Last_iteration_centroids[2][1]=0.75*New_iteration_centroids[0][1]
      Last_iteration_centroids[2][2]=0.75*New_iteration_centroids[0][2]
      Last_iteration_centroids[2][3]=0.75*New_iteration_centroids[0][3]


    elif (len_C_0==0) :
      print("\nCASE: 6 RARE\n")
      Last_iteration_centroids[1]=New_iteration_centroids[1]
      Last_iteration_centroids[2]=New_iteration_centroids[2]

                  Last_iteration_centroids[0][0]=0.5*(New_iteration_centroids[1][0]     +
New_iteration_centroids[2][0])
                  Last_iteration_centroids[0][1]=0.5*(New_iteration_centroids[1][1]     +
New_iteration_centroids[2][1])
                  Last_iteration_centroids[0][2]=0.5*(New_iteration_centroids[1][2]     +
New_iteration_centroids[2][2])
                  Last_iteration_centroids[0][3]=0.5*(New_iteration_centroids[1][3]     +
New_iteration_centroids[2][3])


    elif (len_C_1==0) :
      print("\nCASE: 7 RARE\n")
      Last_iteration_centroids[0]=New_iteration_centroids[0]
      Last_iteration_centroids[2]=New_iteration_centroids[2]

                  Last_iteration_centroids[1][0]=0.5*(New_iteration_centroids[0][0]     +
New_iteration_centroids[2][0])
                  Last_iteration_centroids[1][1]=0.5*(New_iteration_centroids[0][1]     +
New_iteration_centroids[2][1])
                  Last_iteration_centroids[1][2]=0.5*(New_iteration_centroids[0][2]     +
New_iteration_centroids[2][2])
```

```
                  Last_iteration_centroids[1][3]=0.5*(New_iteration_centroids[0][3]     +
New_iteration_centroids[2][3])


    elif (len_C_2==0) :
      print("\nCASE: 8 RARE\n")
      Last_iteration_centroids[0]=New_iteration_centroids[0]
      Last_iteration_centroids[1]=New_iteration_centroids[1]

                  Last_iteration_centroids[2][0]=0.5*(New_iteration_centroids[0][0]     +
New_iteration_centroids[1][0])
                  Last_iteration_centroids[2][1]=0.5*(New_iteration_centroids[0][1]     +
New_iteration_centroids[1][1])
                  Last_iteration_centroids[2][2]=0.5*(New_iteration_centroids[0][2]     +
New_iteration_centroids[1][2])
                  Last_iteration_centroids[2][3]=0.5*(New_iteration_centroids[0][3]     +
New_iteration_centroids[1][3])

     print(f"\nChecking  for  Convergence  in  K-Means  Algorithm  in  Iteration  number  :
{Iterations}")


#***************************************************************************************
****************************************************************************************
**********************************************
#Inputs for k-means algorithm with k=3
#***************************************************************************************
****************************************************************************************
**********************************************

url="https://raw.githubusercontent.com/SaumyenMishraIITP/iris/main/Iris.csv"     #    raw-
github-link
df=pd.read_csv(url)                                                          # converting
csv to dataframe
df=df.drop(['Id'], axis=1)                                                     # removing
the column 'Id'

initial_centroids_set_1=[[mean_setosa_sep_len,mean_setosa_sep_wid,mean_setosa_pet_len,mea
n_setosa_pet_wid],
[mean_versicolor_sep_len,mean_versicolor_sep_wid,mean_versicolor_pet_len,mean_versicolor_
pet_wid],
[mean_virginica_sep_len,mean_virginica_sep_wid,mean_virginica_pet_len,mean_virginica_pet_
wid]]
initial_centroids_set_2=[[5,3.5,1.5,0],[6,3,4.5,1.5],[6.5,3,5.5,2]]
initial_centroids_set_3=[[5.25,3.25,1.75,0.5],[4.75,2.75,1.25,0],[6.5,3,5.5,2]]
initial_centroids_set_4=[[5,3.5,1.5,0],[6.25,3,5,1.75],[7,4,6,3]]
initial_centroids_set_5=[[5,2,2,1],[4,3,3,1],[5,3,4,2]]
initial_centroids_set_6=[[2,1,1,0],[3,2,2,1],[4,2,2,1]]


print(f"\nGiven Dataset : \n\n{df}")
```

```
print("\n\n**************************************************************************************************************************************************************************************************")
print("\n**************************************************************************************************************************************************************************************************")

print(f"\nRandom Centroid List : \n\n{initial_centroids_set_1}")

print("\n\n**************************************************************************************************************************************************************************************************")
print("\n**************************************************************************************************************************************************************************************************")

kmeans_iris(initial_centroids_set_1,df)
```

2. Text Output of K-Means_Iris Python Code for Centroid Set 1:

```
Given Dataset :

     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
0              5.1           3.5            1.4           0.2     Iris-setosa
1              4.9           3.0            1.4           0.2     Iris-setosa
2              4.7           3.2            1.3           0.2     Iris-setosa
3              4.6           3.1            1.5           0.2     Iris-setosa
4              5.0           3.6            1.4           0.2     Iris-setosa
..             ...           ...            ...           ...             ...
145            6.7           3.0            5.2           2.3  Iris-virginica
146            6.3           2.5            5.0           1.9  Iris-virginica
147            6.5           3.0            5.2           2.0  Iris-virginica
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]


******************************************************************************
******************************************************************************
**************************************

******************************************************************************
******************************************************************************
**************************************

Random Centroid List :
```

```
[[5.005999999999999, 3.4180000000000006, 1.464, 0.2439999999999999], [5.936, 2.7700000000000005, 4.26, 1.3259999999999998], [6.587999999999998, 2.9739999999999998, 5.552, 2.026]]

*****************************************************************************
*****************************************************************************
****************************************

*****************************************************************************
*****************************************************************************
****************************************

Iteration Number : 1

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 1

Iteration Number : 2

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 2

Iteration Number : 3

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 3

Iteration Number : 4

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 4

Iteration Number : 5


*****************************************************************************
*****************************************************************************
****************************************

*****************************************************************************
*****************************************************************************
****************************************
```

CASE: 1 COMMON

Convergence Achieved in th K-Means Clustering Algorithm!!!

Total Number of Iterations : 5

Total Number of Clusters : 3

Total Number of Points in First Cluster : 50

SSE for the First Cluster : 15.240400000000003 cm.Sq.

Total Number of Points in Second Cluster : 61

SSE for the Second Cluster : 38.29081967213114 cm.Sq.

Total Number of Points in Third Cluster : 39

SSE for the Third Cluster : 25.413846153846155 cm.Sq.

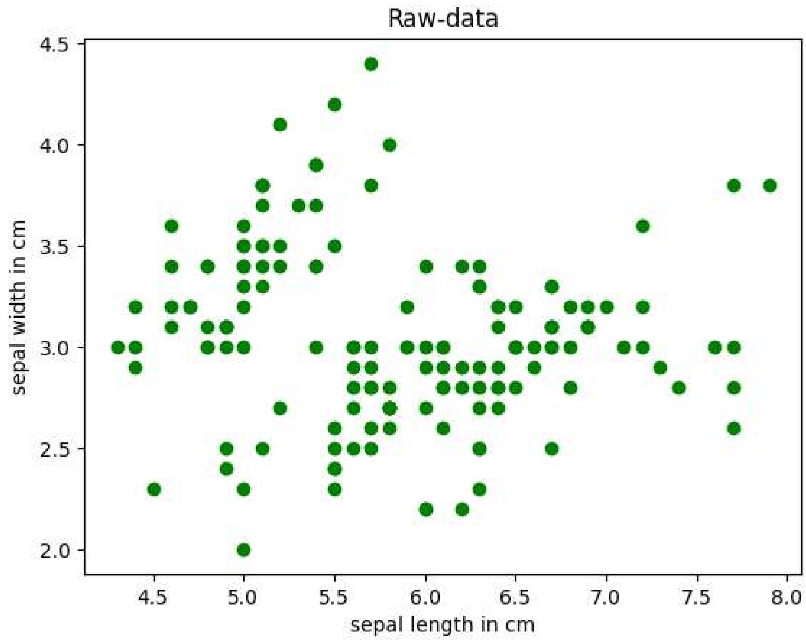
Total SSE for the finalized 3-Cluster-System : 78.9450658259773 cm.Sq.

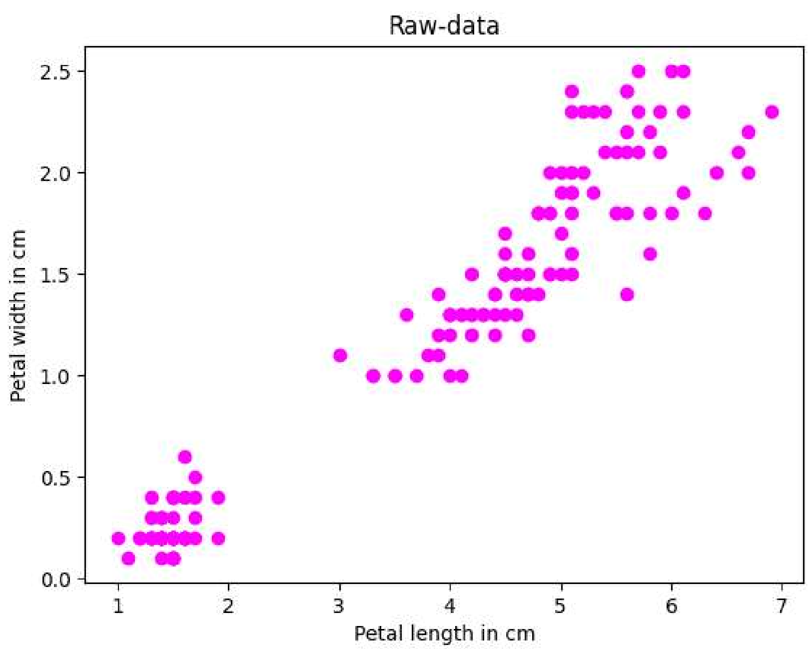
In Cluster 0, the number of elements from :
a.)CLASS SETOSA : 50
b.)CLASS VERSICOLOR : 0
c.)CLASS VIRGINICA : 0
Shannon's Entropy for Cluster 0 is : 0.0

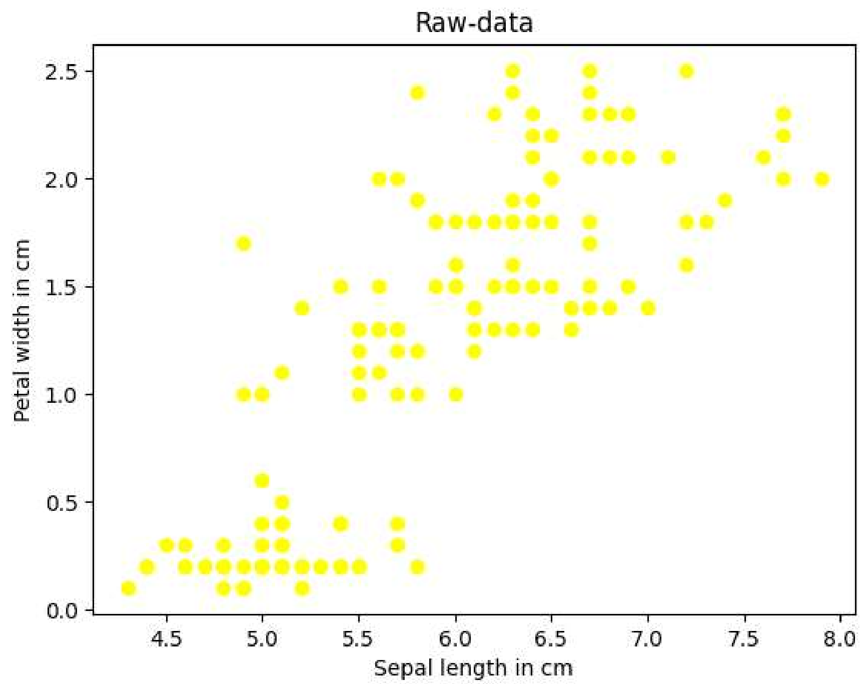
In Cluster 1, the number of elements from :
a.)CLASS SETOSA : 0
b.)CLASS VERSICOLOR : 47
c.)CLASS VIRGINICA : 14
Shannon's Entropy for Cluster 1 is : 0.5981316527720559

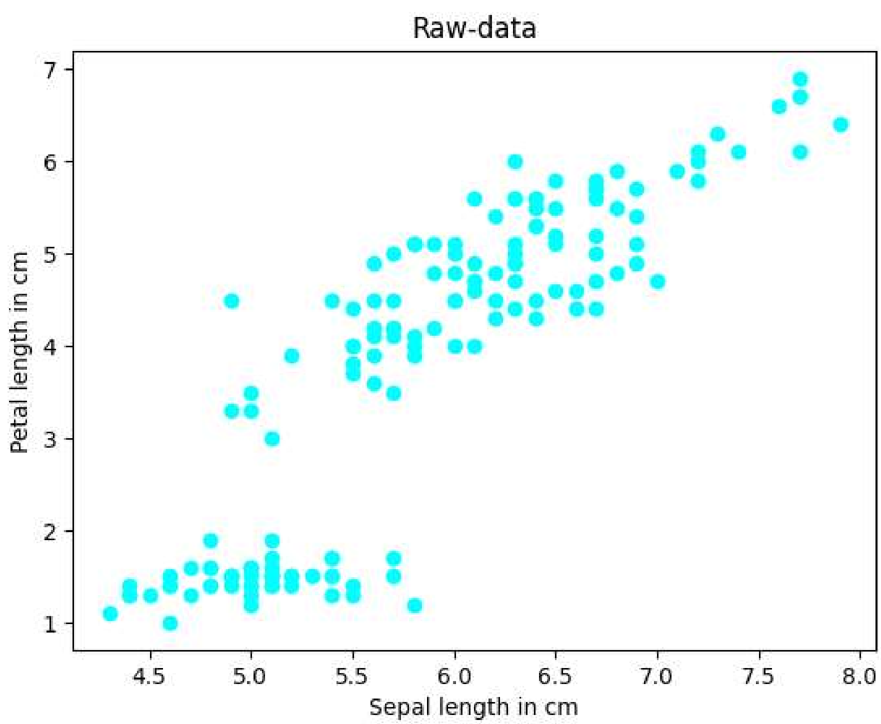
In Cluster 2, the number of elements from :
a.)CLASS SETOSA : 0
b.)CLASS VERSICOLOR : 3
c.)CLASS VIRGINICA : 36
Shannon's Entropy for Cluster 2 is : 0.5847640769425511

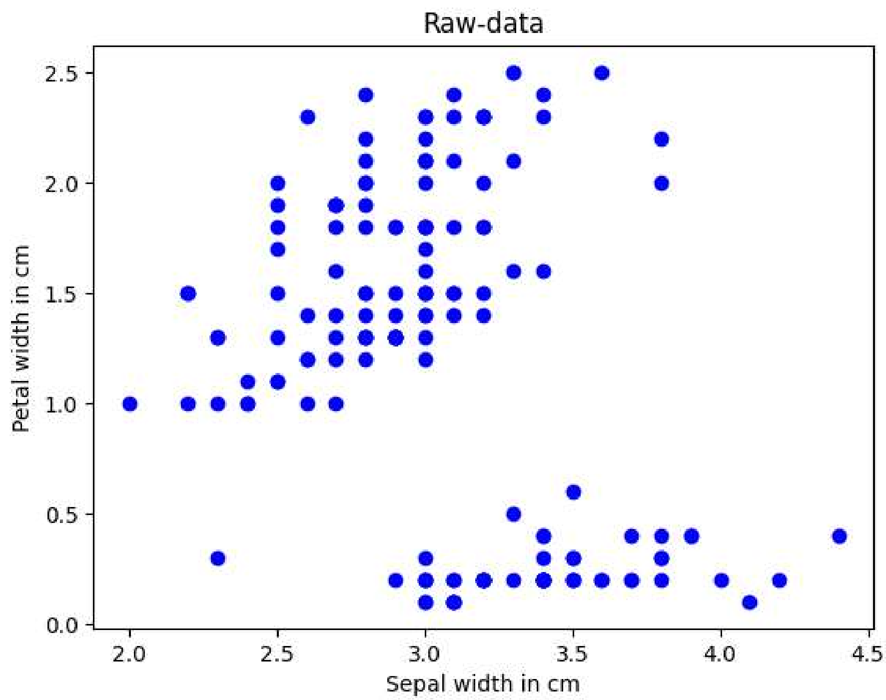
3. Scatter Plots for raw data and Clustered data with Centroid Set 1:

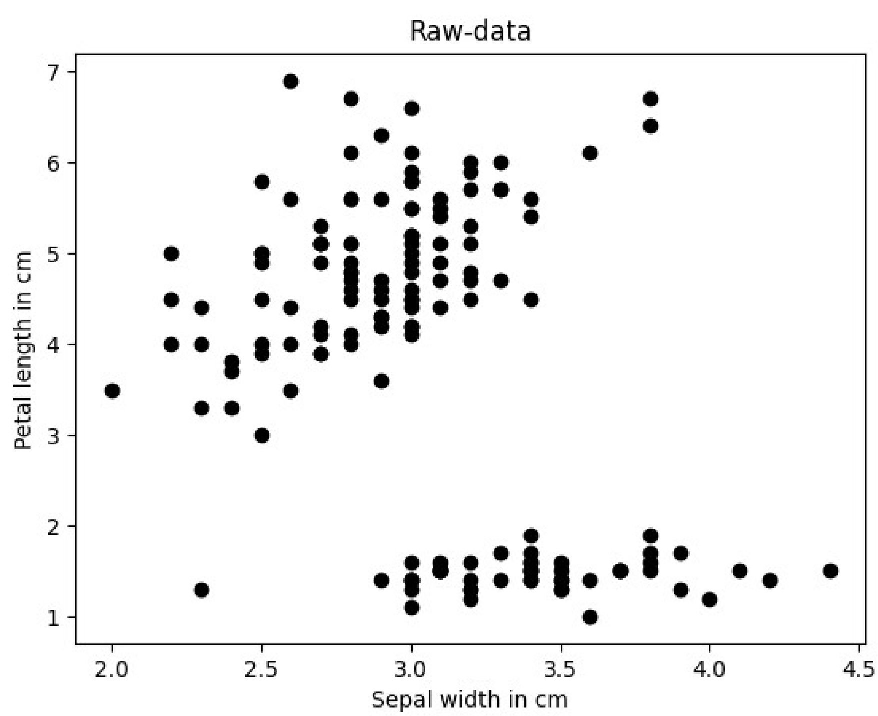


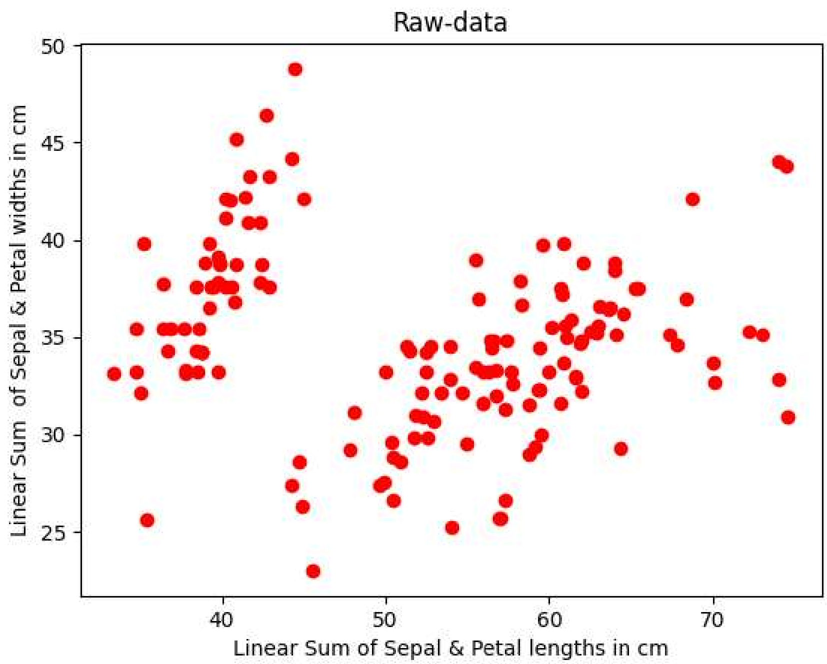


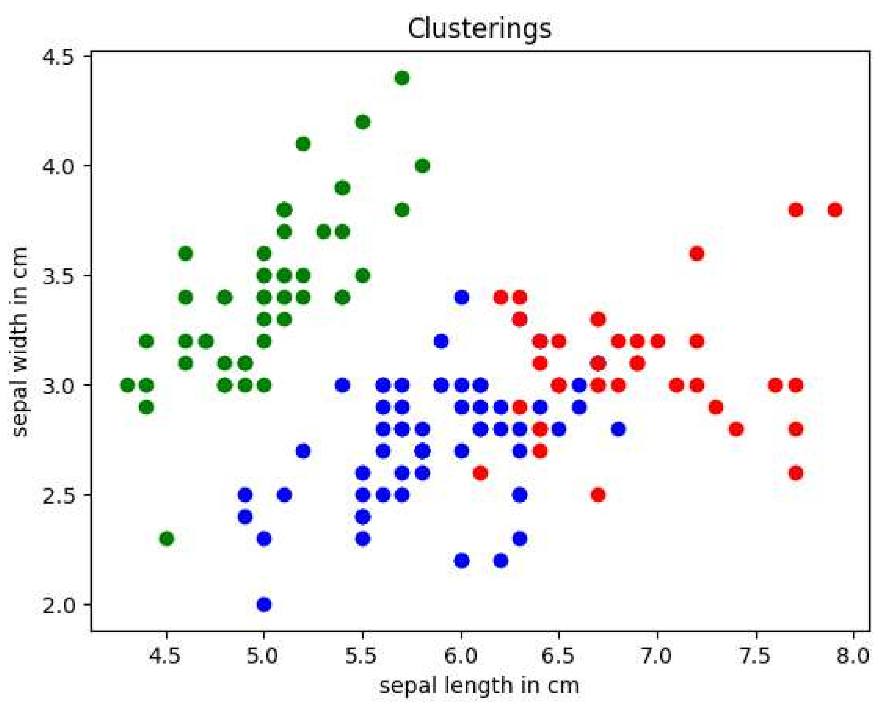

Raw-data
Petal width in cm
Sepal length in cm
0.0
0.5
1.0
1.5
2.0
2.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

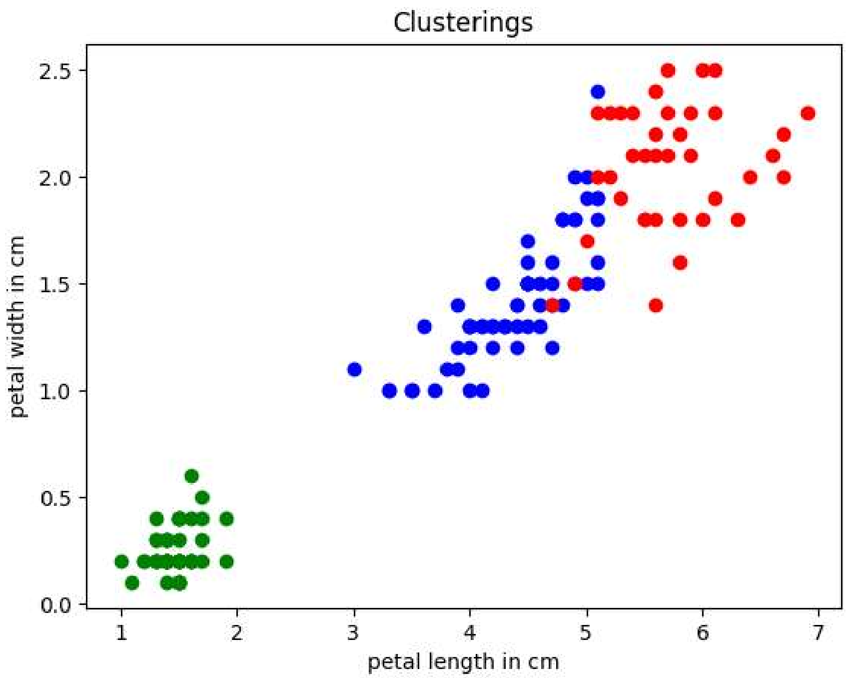

Raw-data
Petal length in cm
Sepal length in cm
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Raw-data
Petal width in cm
Sepal width in cm
0.0
0.5
1.0
1.5
2.0
2.5
2.0
2.5
3.0
3.5
4.0
4.5

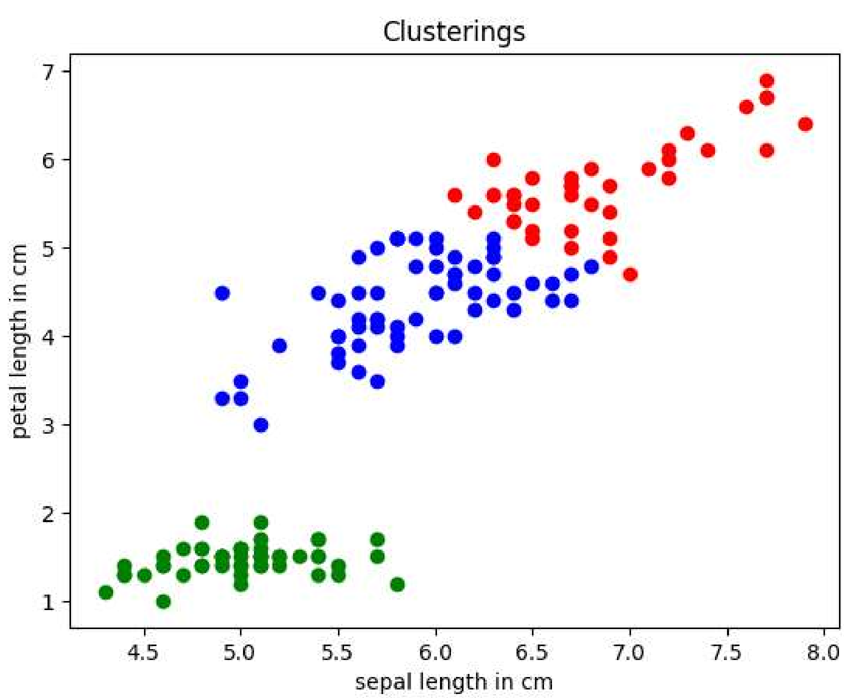

Raw-data
Petal length in cm
Sepal width in cm
1
2
3
4
5
6
7
2.0
2.5
3.0
3.5
4.0
4.5


Raw-data
Linear Sum of Sepal & Petal widths in cm
Linear Sum of Sepal & Petal lengths in cm
50
45
40
35
30
25
40
50
60
70

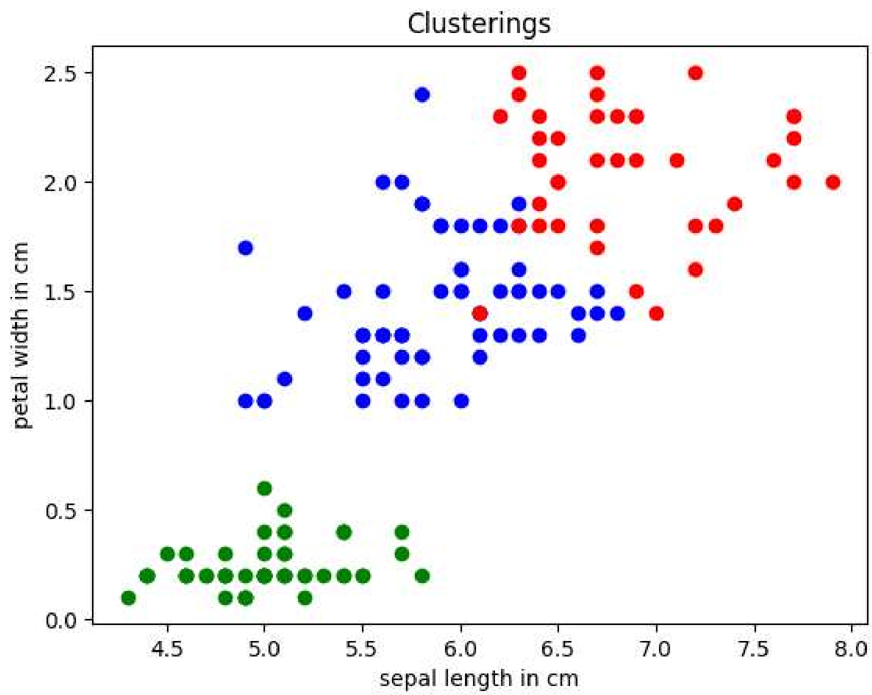

Clusterings
sepal width in cm
sepal length in cm
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

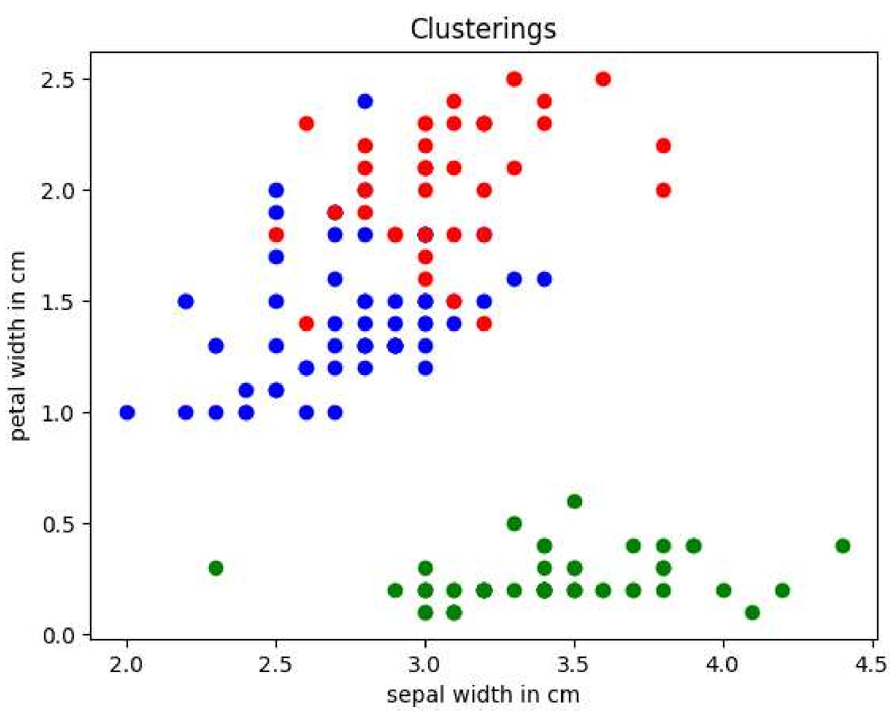


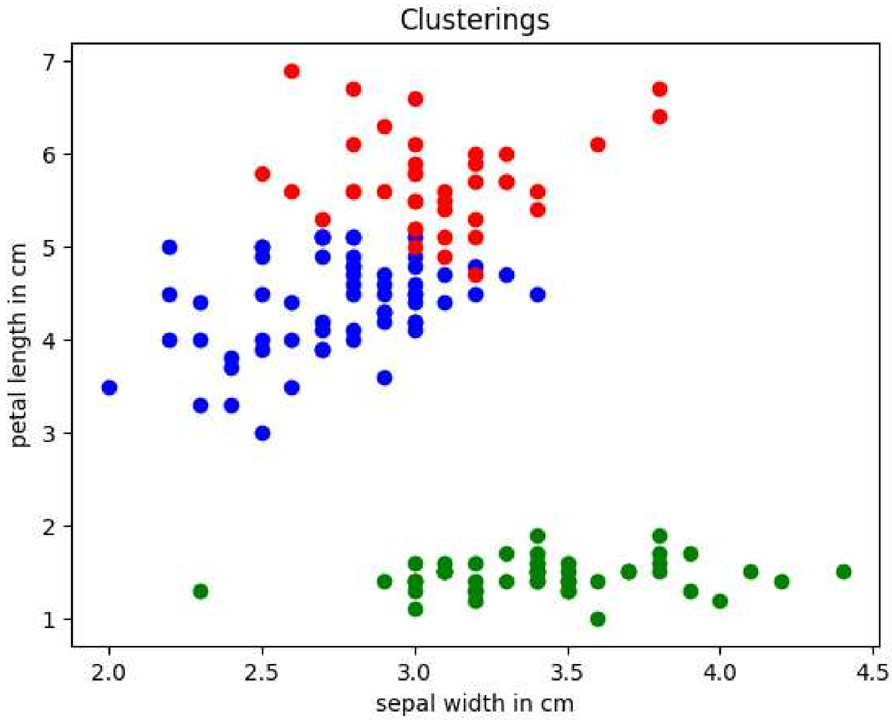
Clusterings
petal width in cm
petal length in cm
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

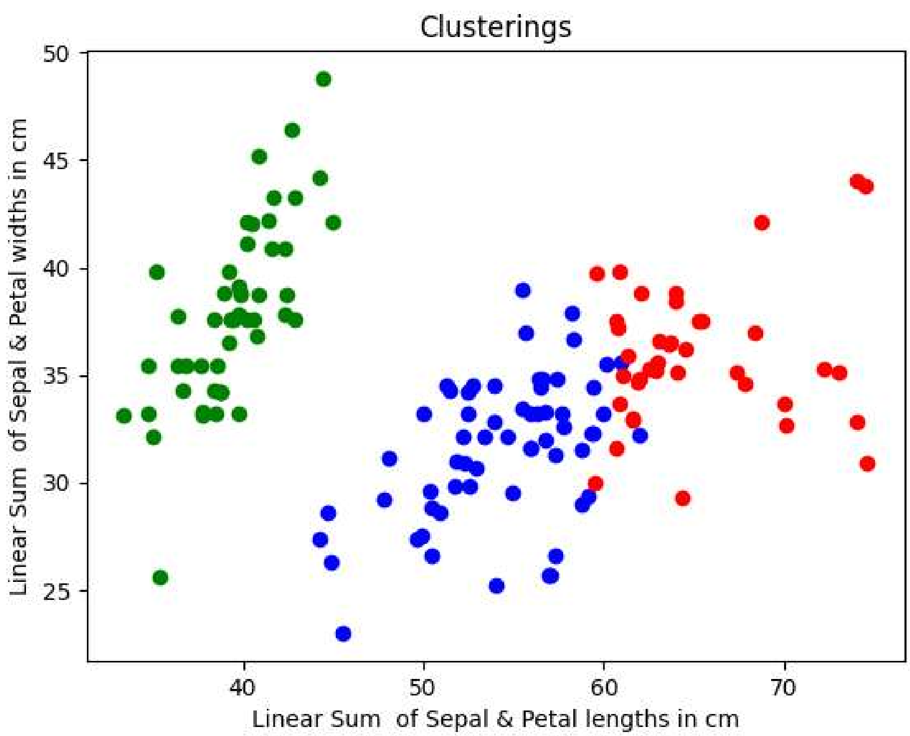


Clusterings
petal length in cm
sepal length in cm
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Clusterings
petal width in cm
sepal length in cm
2.5
2.0
1.5
1.0
0.5
0.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Clusterings
petal width in cm
sepal width in cm
2.5
2.0
1.5
1.0
0.5
0.0
2.0
2.5
3.0
3.5
4.0
4.5


Clusterings
petal length in cm
sepal width in cm
7
6
5
4
3
2
1
2.0
2.5
3.0
3.5
4.0
4.5


Clusterings
Linear Sum of Sepal & Petal widths in cm
Linear Sum of Sepal & Petal lengths in cm
50
45
40
35
30
25
40
50
60
70

4. Text Output, Scatter Plots for raw data and Clustered data with Centroid Set 3:

Given Dataset :

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
0              5.1           3.5            1.4           0.2     Iris-setosa
1              4.9           3.0            1.4           0.2     Iris-setosa
2              4.7           3.2            1.3           0.2     Iris-setosa
3              4.6           3.1            1.5           0.2     Iris-setosa
4              5.0           3.6            1.4           0.2     Iris-setosa
..             ...           ...            ...           ...             ...
145            6.7           3.0            5.2           2.3  Iris-virginica
146            6.3           2.5            5.0           1.9  Iris-virginica
147            6.5           3.0            5.2           2.0  Iris-virginica
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]
```

```
**************************************************************************
**************************************************************************
**********************************************

**************************************************************************
**************************************************************************
**********************************************
```

Random Centroid List :

[[5.25, 3.25, 1.75, 0.5], [4.75, 2.75, 1.25, 0], [6.5, 3, 5.5, 2]]

```
**************************************************************************
**************************************************************************
**********************************************

**************************************************************************
**************************************************************************
**********************************************

Iteration Number : 1

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 1

Iteration Number : 2

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 2
```

Iteration Number : 3

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 3

Iteration Number : 4

CASE: 2 COMMON

Checking for Convergence in K-Means Algorithm in Iteration number : 4

Iteration Number : 5

****
****
****

****
****
****

CASE: 1 COMMON

Convergence Achieved in th K-Means Clustering Algorithm!!!

Total Number of Iterations : 5

Total Number of Clusters : 3

Total Number of Points in First Cluster : 31

SSE for the First Cluster : 18.63935483870968 cm.Sq.

Total Number of Points in Second Cluster : 22

SSE for the Second Cluster : 2.84409090909091 cm.Sq.

Total Number of Points in Third Cluster : 97

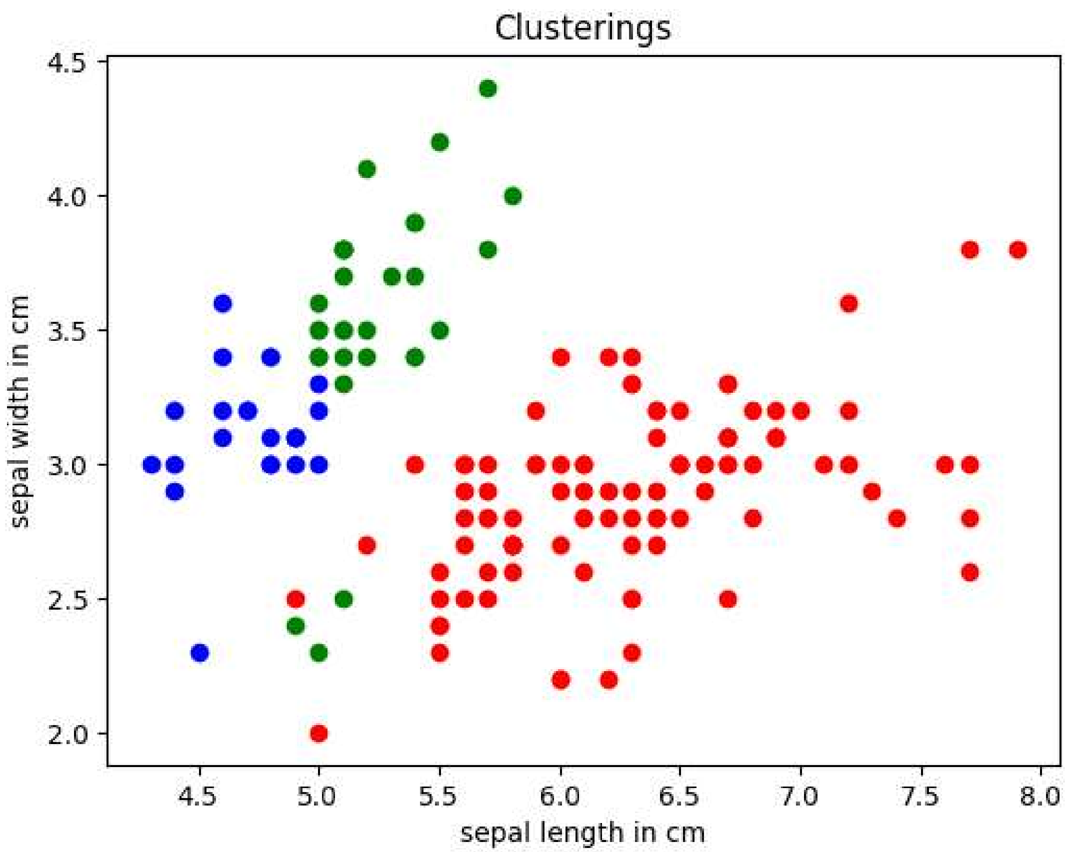
SSE for the Third Cluster : 123.7958762886598 cm.Sq.

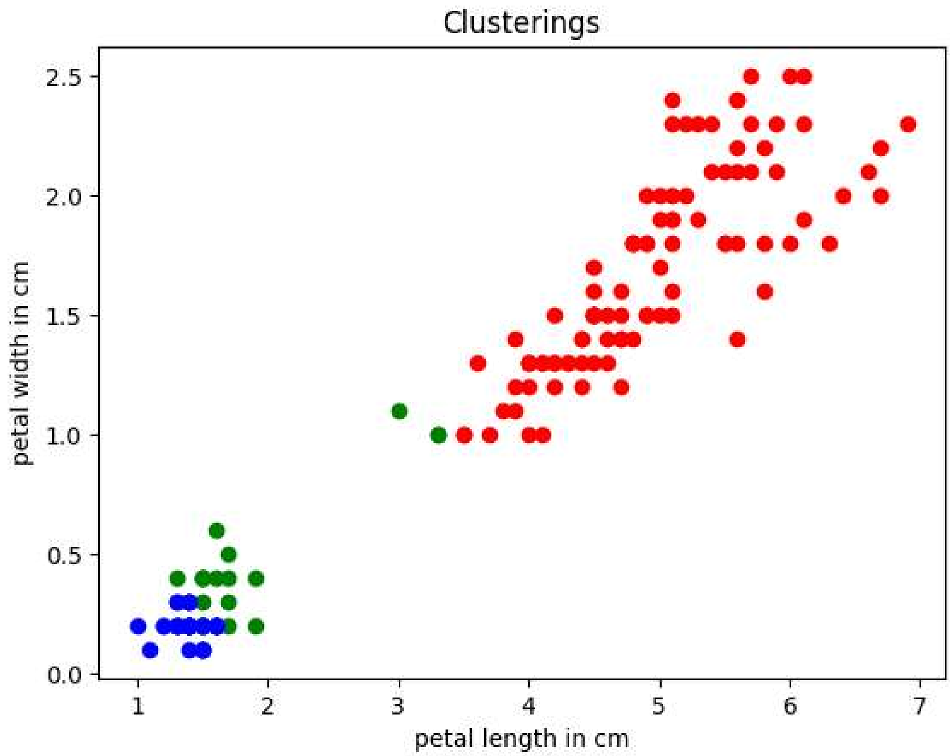
Total SSE for the finalized 3-Cluster-System : 145.27932203646037 cm.Sq.

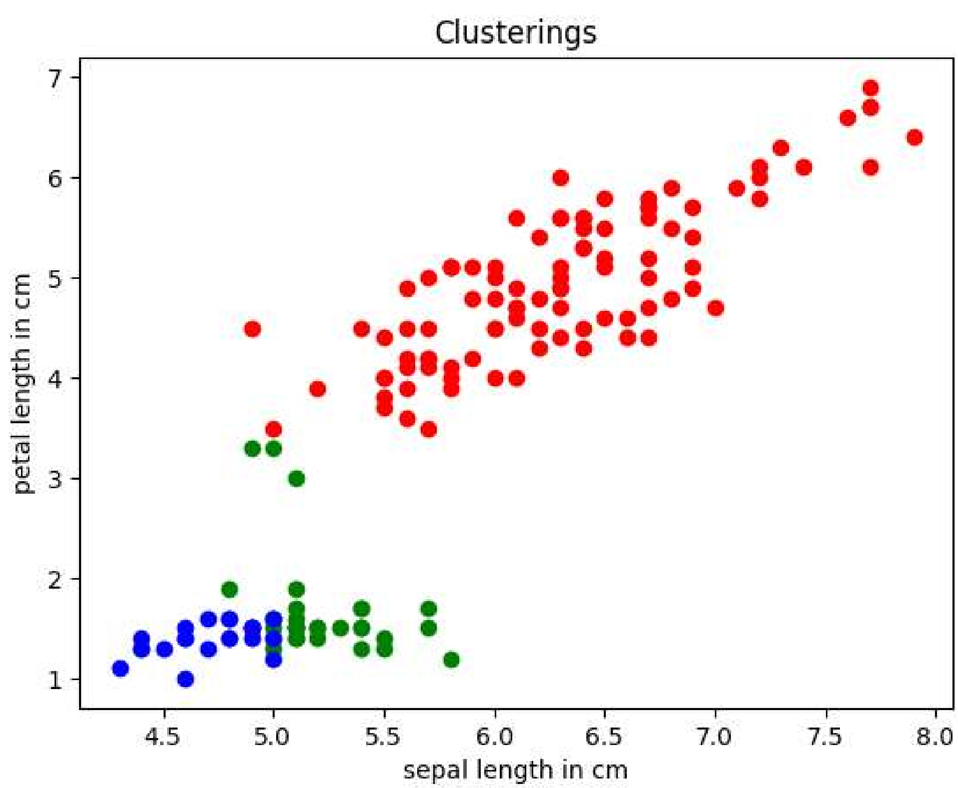
```
In Cluster 0, the number of elements from :
a.)CLASS SETOSA : 28
b.)CLASS VERSICOLOR : 3
c.)CLASS VIRGINICA : 0
Shannon's Entropy for Cluster 0 is : 0.7119743312648016
```

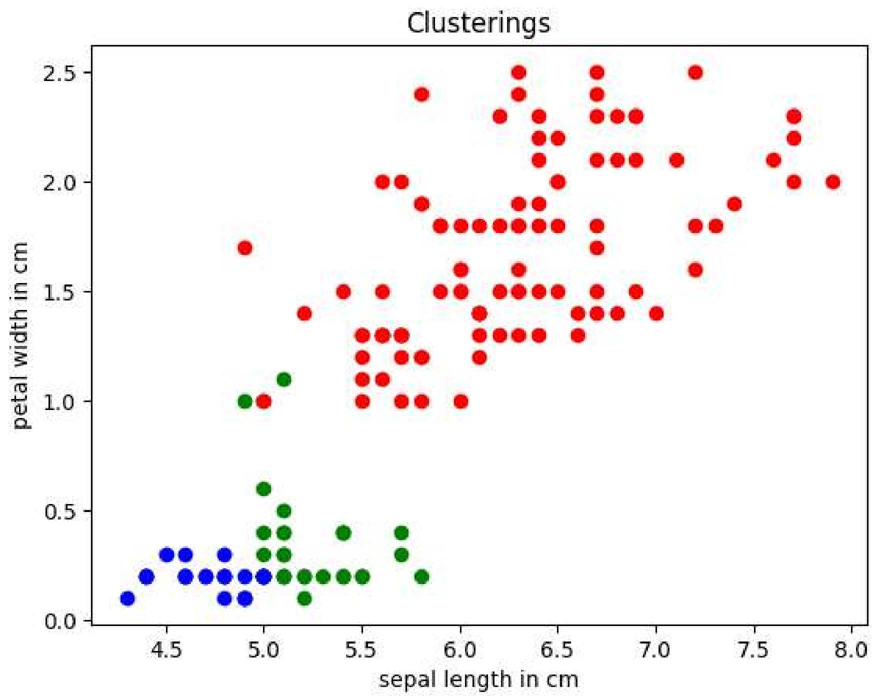
```
In Cluster 1, the number of elements from :
a.)CLASS SETOSA : 22
b.)CLASS VERSICOLOR : 0
c.)CLASS VIRGINICA : 0
Shannon's Entropy for Cluster 1 is : 0.5211468113004681
```

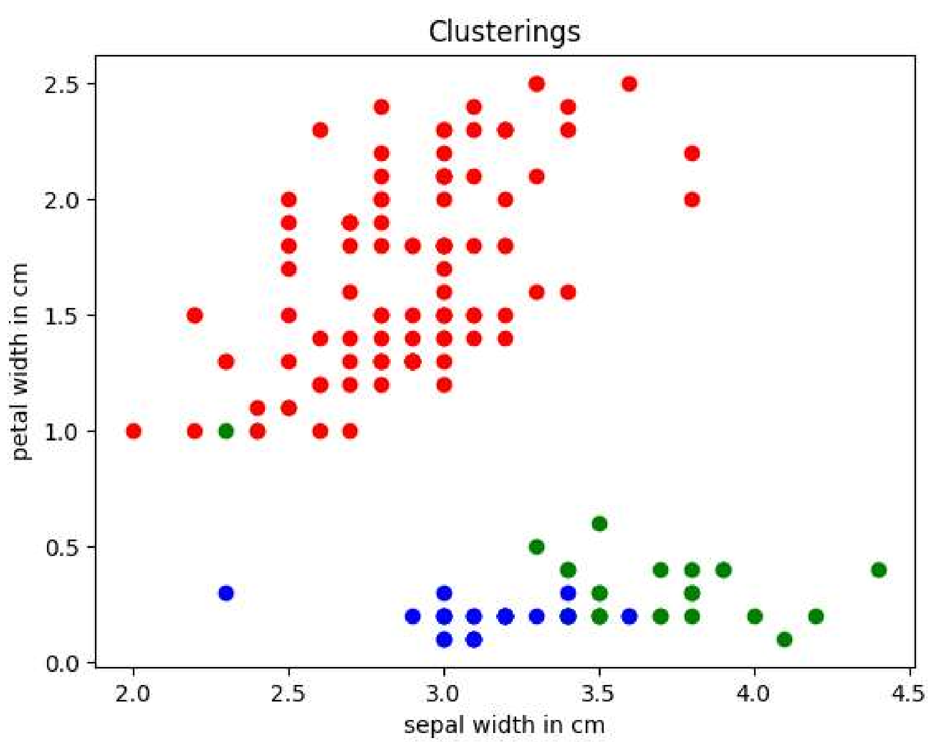
```
In Cluster 2, the number of elements from :
a.)CLASS SETOSA : 0
b.)CLASS VERSICOLOR : 47
c.)CLASS VIRGINICA : 50
Shannon's Entropy for Cluster 2 is : 0.08391129781126216
```

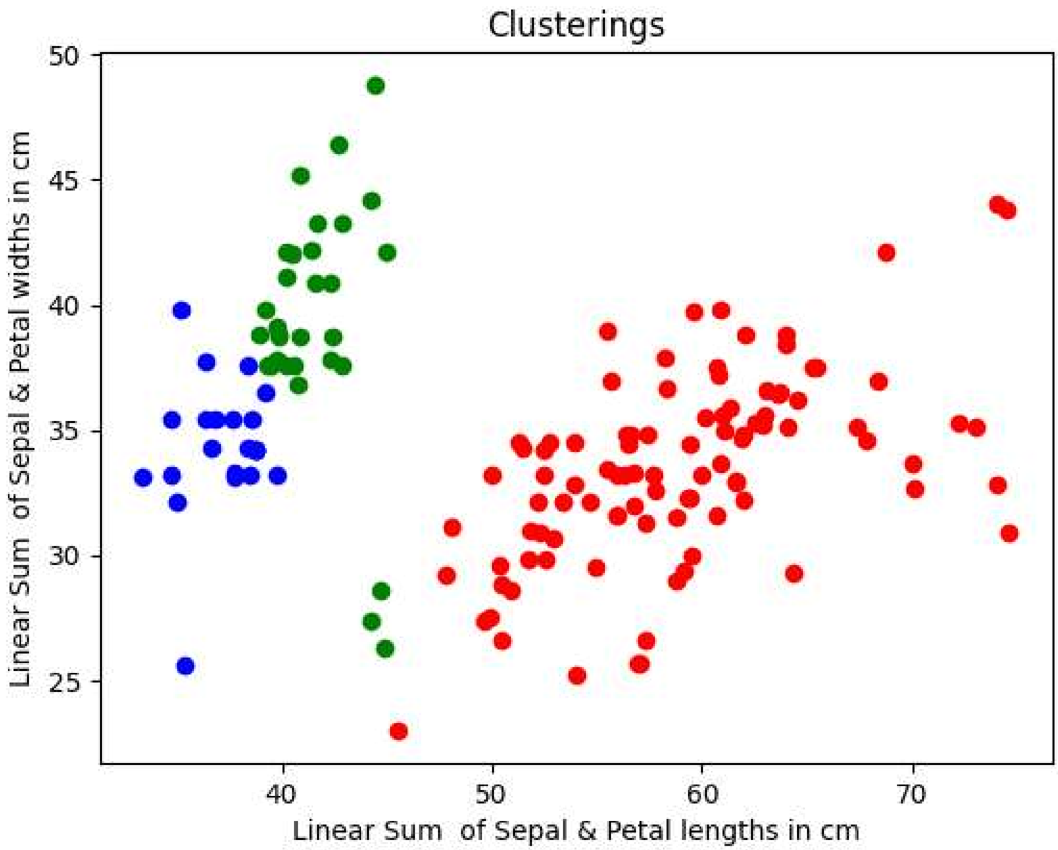



Clusterings
petal width in cm
petal length in cm
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


Clusterings
petal length in cm
sepal length in cm
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Clusterings
2.5
2.0
1.5
1.0
0.5
0.0
petal width in cm
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal length in cm


Clusterings
2.5
2.0
1.5
1.0
0.5
0.0
petal width in cm
2.0
2.5
3.0
3.5
4.0
4.5
sepal width in cm

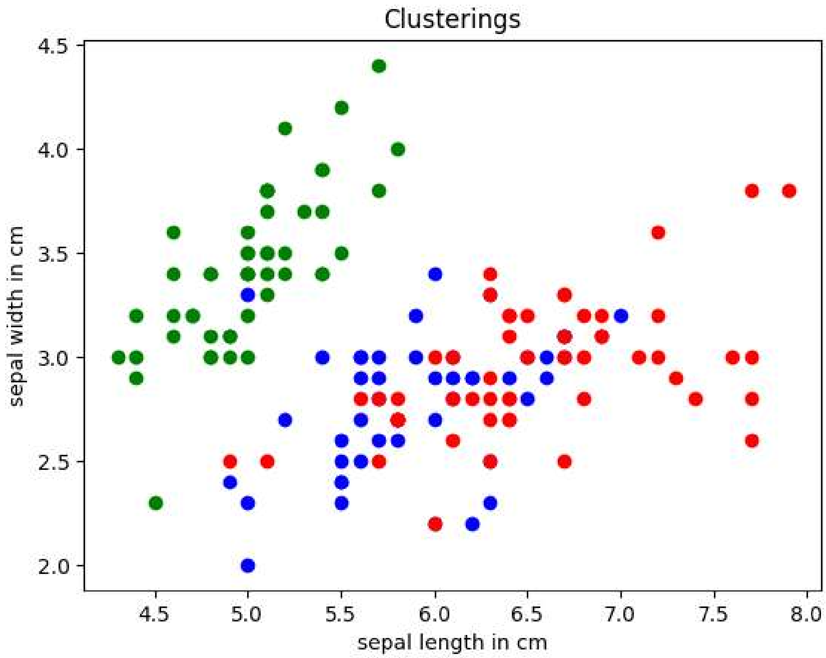

Clusterings
50
45
40
35
30
25
Linear Sum of Sepal & Petal widths in cm
40
50
60
70
Linear Sum of Sepal & Petal lengths in cm

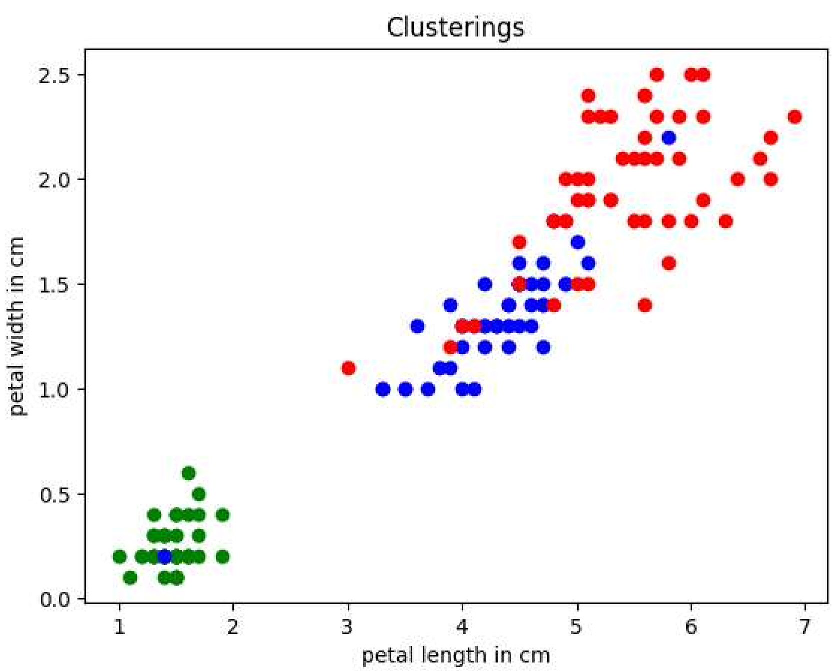
5. K-Medoids Outputs (with medoid_set1) :

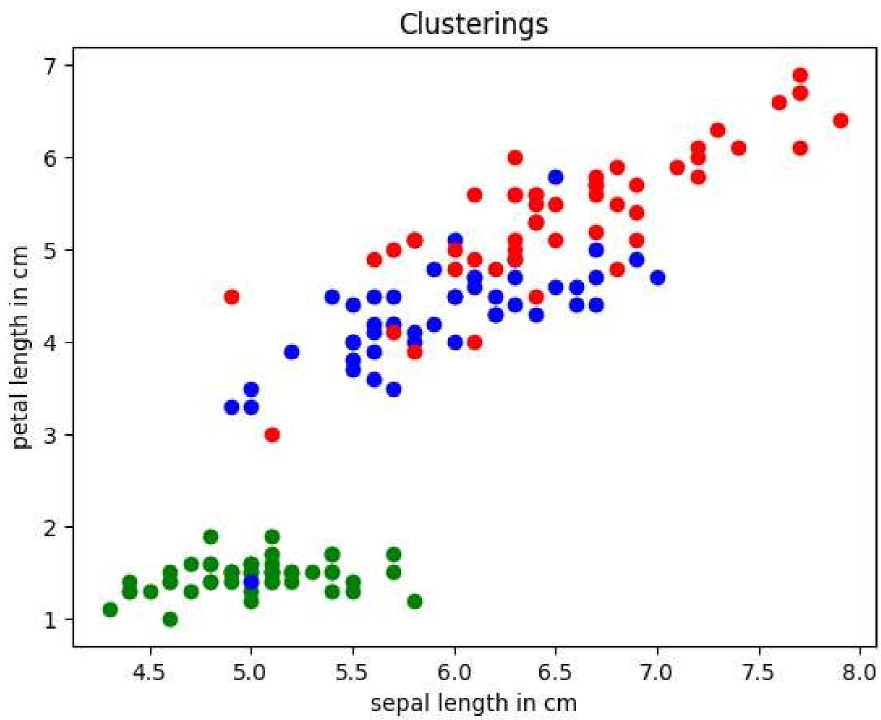
No. of Points : [50,47,53]

SSE : [15.4, 44.7, 68]

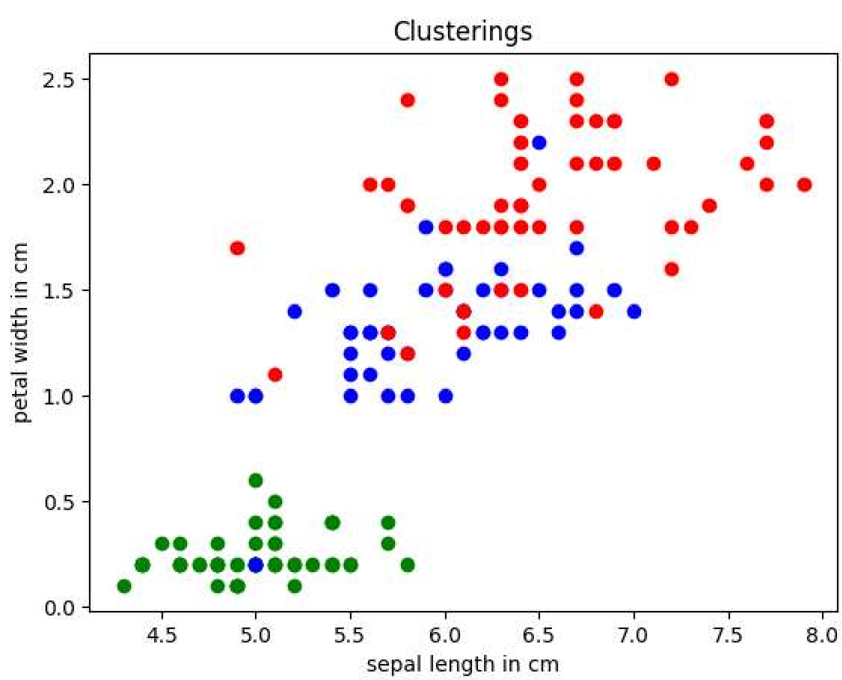
SSE_Total = 128.13



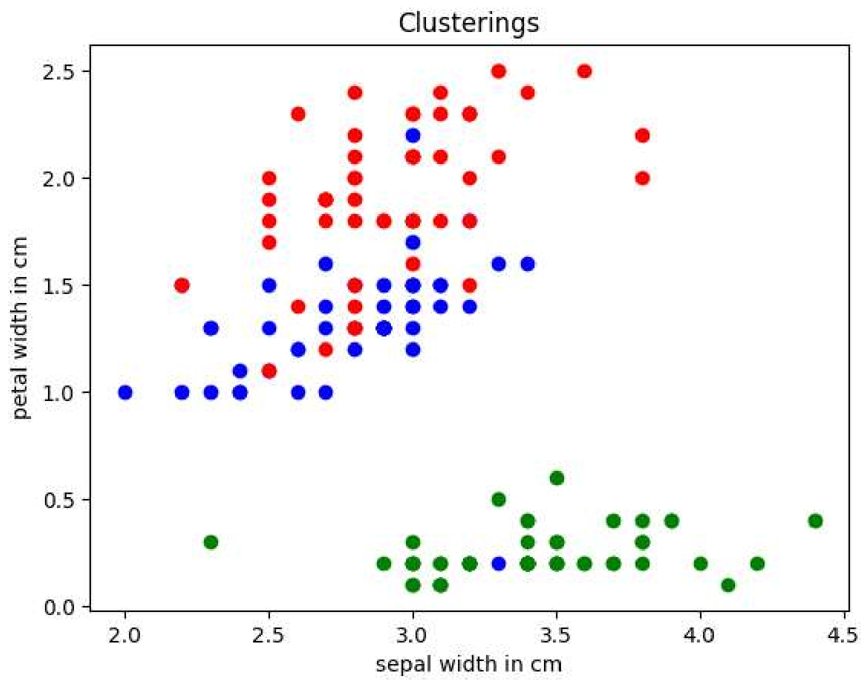




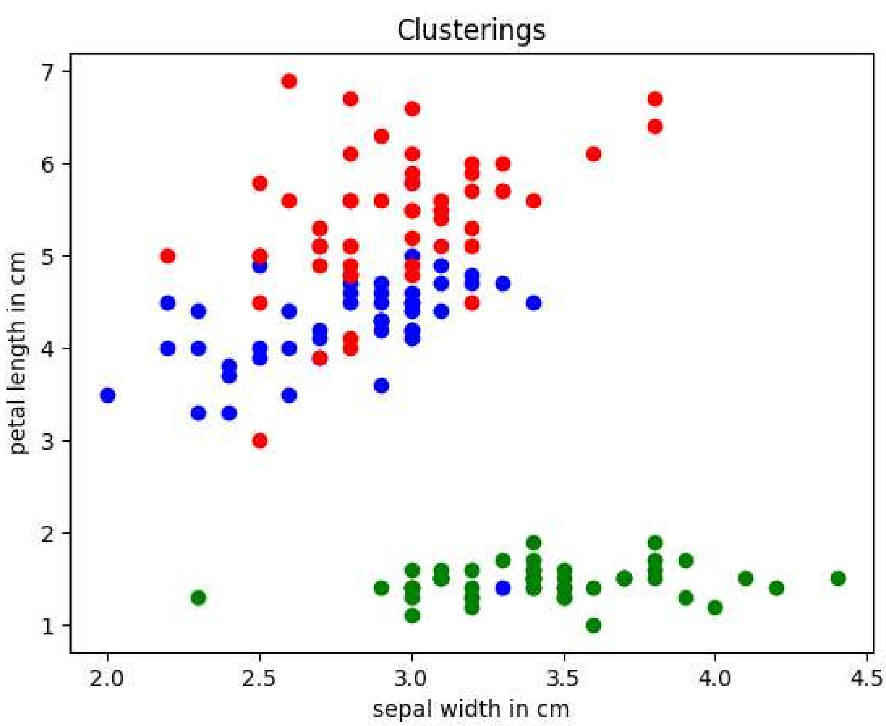
Clusterings
petal length in cm
sepal length in cm
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



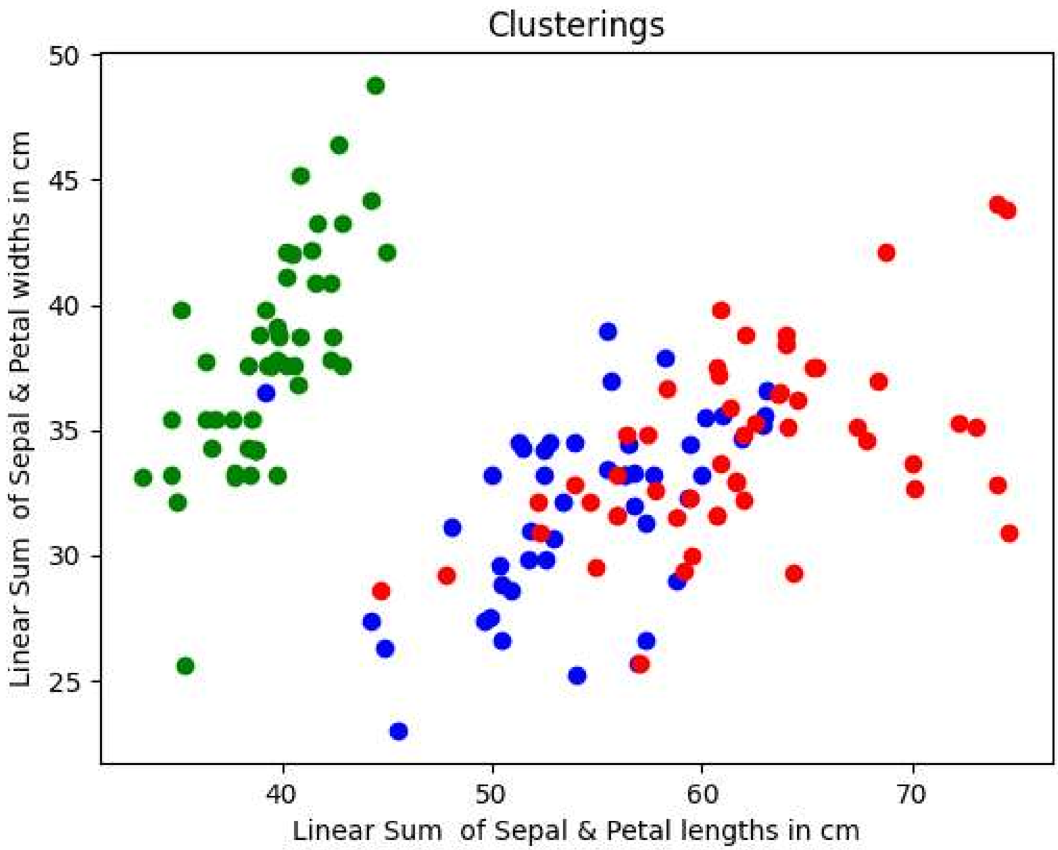
Clusterings
petal width in cm
sepal length in cm
0.0
0.5
1.0
1.5
2.0
2.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



Clusterings
petal width in cm
sepal width in cm



Clusterings
petal length in cm
sepal width in cm



6. Python-Code for K-Medoids :

```
#************************************************************************************************
*************************************************************************************************
************************
#Importing CSV & Converting to dataframe
#************************************************************************************************
*************************************************************************************************
************************

import pandas as pd
import matplotlib.pyplot as plt
import math

url="https://raw.githubusercontent.com/SaumyenMishraIITP/iris/main/Iris.csv" # raw-github-link
df=pd.read_csv(url)                                                             # converting csv to
dataframe
df=df.drop(['Id'], axis=1)                                                      # removing the column
'Id'

#print (df)
#print(df.index[0])
#print(df.columns[3])
```

```
tot_rows=len(df.index)                                                        # Counting no. of rows
in dataframe
tot_cols=len(df.columns)                                                      # Counting no. of
columns in dataframe

#****************************************************************************************************
****************************************************************************************************
*************************
#Exploratory data analysis & initialization of Medoids
#****************************************************************************************************
****************************************************************************************************
*************************

n_setosa=0
n_versicolor=0
n_virginica=0

kmeans = [[5.005999999999999, 3.4180000000000006, 1.464, 0.2439999999999999], [5.88360655737705,
2.740983606557377, 4.388524590163935, 1.4344262295081966], [6.853846153846153, 3.0769230769230766,
5.715384615384615, 2.053846153846153]]

k11=0
k21=0
k31=0

dist11=[]
dist21=[]
dist31=[]

for i1 in range(0,tot_rows,1) :
    k11=math.sqrt(((kmeans[0][0] - df.iat[i1,0])**2) + ((kmeans[0][1] - df.iat[i1,1])**2) +
((kmeans[0][2] - df.iat[i1,2])**2) + ((kmeans[0][3] - df.iat[i1,3])**2))
    dist11.append(k11)

    k21=math.sqrt(((kmeans[1][0] - df.iat[i1,0])**2) + ((kmeans[1][1] - df.iat[i1,1])**2) +
((kmeans[1][2] - df.iat[i1,2])**2) + ((kmeans[1][3] - df.iat[i1,3])**2))
    dist21.append(k21)

    k31=math.sqrt(((kmeans[2][0] - df.iat[i1,0])**2) + ((kmeans[2][1] - df.iat[i1,1])**2) +
((kmeans[2][2] - df.iat[i1,2])**2) + ((kmeans[2][3] - df.iat[i1,3])**2))
    dist31.append(k31)

MIN_D1=min(dist11)
MIN_D2=min(dist21)
MIN_D3=min(dist31)


IND_D1=dist11.index(MIN_D1)
IND_D2=dist21.index(MIN_D2)
IND_D3=dist31.index(MIN_D3)



for i in range(0,tot_rows,1) :
  if  (df.iat[i,4]=="Iris-setosa") :
```

```
    df.iat[i,4] = 0
    n_setosa = n_setosa+1

  elif  (df.iat[i,4]=="Iris-versicolor") :
    df.iat[i,4] = 1
    n_versicolor = n_versicolor+1

  elif (df.iat[i,4]=="Iris-virginica") :
      df.iat[i,4] = 2
      n_virginica = n_virginica+1
print(n_setosa)
print(n_versicolor)
print(n_virginica)
kmedoids=[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]
kmedoids[0] = [df.iat[IND_D1,0], df.iat[IND_D1,1], df.iat[IND_D1,2], df.iat[IND_D1,3],
df.iat[IND_D1,4]]
kmedoids[1] = [df.iat[IND_D2,0], df.iat[IND_D2,1], df.iat[IND_D2,2], df.iat[IND_D2,3],
df.iat[IND_D2,4]]
kmedoids[2] = [df.iat[IND_D3,0], df.iat[IND_D3,1], df.iat[IND_D3,2], df.iat[IND_D3,3],
df.iat[IND_D3,4]]
#print(kmedoids)
print (df)

#*****************************************************************************************************
******************************************************************************************************
*************************
#SCATTER PLOT OF ORIGINAL DATASET
#*****************************************************************************************************
******************************************************************************************************
*************************

sepallen=[]
sepalwid=[]
petallen=[]
petalwid=[]
sepallen_X_petallen=[]
sepalwid_X_petalwid=[]

for p in range(0, len(df.index), 1) :
  sepallen.append(df.iat[p,0])
  sepalwid.append(df.iat[p,1])
  petallen.append(df.iat[p,2])
  petalwid.append(df.iat[p,3])
  sepallen_X_petallen.append(((7*df.iat[p,0]) + (3*df.iat[p,2])))
  sepalwid_X_petalwid.append(((11*(df.iat[p,1])) + (df.iat[p,3])))

plt.scatter(sepallen,sepalwid, c='green')
plt.xlabel('sepal length in cm')
plt.ylabel('sepal width in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(petallen,petalwid,c='magenta')
plt.xlabel('Petal length in cm')
plt.ylabel('Petal width in cm')
plt.title('Raw-data')
```

```
plt.show()

plt.scatter(sepallen,petallen,c='cyan')
plt.xlabel('Sepal length in cm')
plt.ylabel('Petal length in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepallen,petalwid,c='yellow')
plt.xlabel('Sepal length in cm')
plt.ylabel('Petal width in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepalwid,petalwid,c='blue')
plt.xlabel('Sepal width in cm')
plt.ylabel('Petal width in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepalwid,petallen,c='black')
plt.xlabel('Sepal width in cm')
plt.ylabel('Petal length in cm')
plt.title('Raw-data')
plt.show()

plt.scatter(sepallen_X_petallen,sepalwid_X_petalwid,c='red')
plt.xlabel('Linear Sum of Sepal & Petal lengths in cm')
plt.ylabel('Linear Sum  of Sepal & Petal widths in cm')
plt.title('Raw-data')
plt.show()


#***************************************************************************************************************************************************************************************************
#Medoid Function, SSE calculation Function, K-Medoids Assignment function, K-Medoids-Iris Function
#***************************************************************************************************************************************************************************************************

def assignment_function(medoids,dataset) :

  num_rows = len(dataset.index)
  num_cols = len(dataset.columns)

  k1=0
  k2=0
  k3=0

  dist1=[]
  dist2=[]
  dist3=[]


  C0_m=0
```

```
  C1_m=0
  C2_m=0


  for t in range (0, num_rows, 1) :
    if (dataset.iat[t,0]==medoids[0][0] and dataset.iat[t,1]==medoids[0][1] and
dataset.iat[t,2]==medoids[0][2] and dataset.iat[t,3]==medoids[0][3] and
dataset.iat[t,4]==medoids[0][4]) :
      C0_m=t
      #print(C0_m)
    elif (dataset.iat[t,0]==medoids[1][0] and dataset.iat[t,1]==medoids[1][1] and
dataset.iat[t,2]==medoids[1][2] and dataset.iat[t,3]==medoids[1][3] and
dataset.iat[t,4]==medoids[1][4]) :
      C1_m=t
      #print(C1_m)
    elif (dataset.iat[t,0]==medoids[2][0] and dataset.iat[t,1]==medoids[2][1] and
dataset.iat[t,2]==medoids[2][2] and dataset.iat[t,3]==medoids[2][3] and
dataset.iat[t,4]==medoids[2][4]) :
      C2_m=t
      #print(C2_m)


  for i in range(0,num_rows,1) :
    if (i!=C0_m and i!=C1_m and i!=C2_m) :
        k1=math.sqrt(((medoids[0][0] - dataset.iat[i,0])**2) + ((medoids[0][1] -
dataset.iat[i,1])**2) + ((medoids[0][2] - dataset.iat[i,2])**2) + ((medoids[0][3] -
dataset.iat[i,3])**2))
        dist1.append(k1)

        k2=math.sqrt(((medoids[1][0] - dataset.iat[i,0])**2) + ((medoids[1][1] -
dataset.iat[i,1])**2) + ((medoids[1][2] - dataset.iat[i,2])**2) + ((medoids[1][3] -
dataset.iat[i,3])**2))
        dist2.append(k2)

        k3=math.sqrt(((medoids[2][0] - dataset.iat[i,0])**2) + ((medoids[2][1] -
dataset.iat[i,1])**2) + ((medoids[2][2] - dataset.iat[i,2])**2) + ((medoids[2][3] -
dataset.iat[i,3])**2))
        dist3.append(k3)

  list0=[]
  list1=[]
  list2=[]

  for j in range (0,num_rows-3,1) :
    if (min(dist1[j],dist2[j],dist3[j])==dist1[j]) :
      a0_0=dataset.iat[j,0]
      a0_1=dataset.iat[j,1]
      a0_2=dataset.iat[j,2]
      a0_3=dataset.iat[j,3]
      a0_4=dataset.iat[j,4]
      a0=[a0_0, a0_1, a0_2, a0_3, a0_4]
      list0.append(a0)

    elif (min(dist1[j],dist2[j],dist3[j])==dist2[j]) :
      a1_0=dataset.iat[j,0]
      a1_1=dataset.iat[j,1]
```

```
      a1_2=dataset.iat[j,2]
      a1_3=dataset.iat[j,3]
      a1_4=dataset.iat[j,4]
      a1=[a1_0, a1_1, a1_2, a1_3,a1_4]
      list1.append(a1)

    elif (min(dist1[j],dist2[j],dist3[j])==dist3[j]) :
      a2_0=dataset.iat[j,0]
      a2_1=dataset.iat[j,1]
      a2_2=dataset.iat[j,2]
      a2_3=dataset.iat[j,3]
      a2_4=dataset.iat[j,4]
      a2=[a2_0, a2_1, a2_2, a2_3, a2_4]
      list2.append(a2)

  if (medoids[0][4]==0) :
    list0.append(medoids[0])
  elif(medoids[1][4]==0) :
    list0.append(medoids[1])
  elif(medoids[2][4]==0) :
    list0.append(medoids[2])

  if (medoids[0][4]==1) :
    list1.append(medoids[0])
  elif(medoids[1][4]==1) :
    list1.append(medoids[1])
  elif(medoids[2][4]==1) :
    list1.append(medoids[2])


  if (medoids[0][4]==2) :
    list2.append(medoids[0])
  elif(medoids[1][4]==2) :
    list2.append(medoids[1])
  elif(medoids[2][4]==2) :
    list2.append(medoids[2])

  print(list2)
  cluster0 = pd.DataFrame (list(list0),
columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm","Species"])
  print ("In assignment function, Cluster 0 is calculated")

  cluster1 = pd.DataFrame (list(list1),
columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm","Species"])
  print ("In assignment function, Cluster 1 is calculated")

  cluster2 = pd.DataFrame (list(list2),
columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm","Species"])
  print ("In assignment function, Cluster 2 is calculated")

  len_C0=len(cluster0.index)
  len_C1=len(cluster1.index)
  len_C2=len(cluster2.index)

  SSE_C0=0
  SSE_C1=0
```

```
  SSE_C2=0

  SSE_Clusterings=0

  for i in range (0,len_C0,1) :
      SSE_C0 = SSE_C0 + (((medoids[0][0]-cluster0.iat[i,0])**2) + ((medoids[0][1]-
cluster0.iat[i,1])**2) + ((medoids[0][2]-cluster0.iat[i,2])**2) + ((medoids[0][3]-
cluster0.iat[i,3])**2))

  for i in range (0,len_C1,1) :
      SSE_C1 = SSE_C1 + (((medoids[1][0]-cluster1.iat[i,0])**2) + ((medoids[1][1]-
cluster1.iat[i,1])**2) + ((medoids[1][2]-cluster1.iat[i,2])**2) + ((medoids[1][3]-
cluster1.iat[i,3])**2))

  for i in range (0,len_C2,1) :
      SSE_C2 = SSE_C2 + (((medoids[2][0]-cluster2.iat[i,0])**2) + ((medoids[2][1]-
cluster2.iat[i,1])**2) + ((medoids[2][2]-cluster2.iat[i,2])**2) + ((medoids[2][3]-
cluster2.iat[i,3])**2))

  SSE_Clusterings = SSE_C0 + SSE_C1 + SSE_C2
  #print(f"\n\nSSE_C0 : {SSE_C0}\n\nSSE_C1 : {SSE_C1}\n\nSSE_C2 : {SSE_C2}\n\nSSE_Clusterings :
{SSE_Clusterings}\n\n")

  #print(cluster0)
  #print(cluster1)
  #print(cluster2)

  return cluster0, cluster1, cluster2, SSE_C0, SSE_C1, SSE_C2, SSE_Clusterings, medoids

#*******************************************************************************************************
*******************************************************************************************************
****************************
#*******************************************************************************************************
*******************************************************************************************************
****************************

def updation_function (C0,C1,C2,SSE0,SSE1,SSE2,SSE_tot,Parent_Medoids) :

  len_C0=len(C0.index)
  len_C1=len(C1.index)
  len_C2=len(C2.index)

  print(f"No. of points in Cluster 0 right now : {len_C0}\nNo. of points in Cluster 1 right now :
{len_C1}\nNo. of  points in Cluster 2 right now : {len_C2}\n")

  new_iteration_medoids=[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]

  random_medoids=[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]

  Conv=0

  list_int_1=[]
  X_1=[]

  for i in range (0, tot_rows, 1) :
```

```
    random_medoids=[Parent_Medoids[0],Parent_Medoids[1], [df.iat[i,0], df.iat[i,1], df.iat[i,2],
df.iat[i,3], df.iat[i,4]]]
    list_int_1.append(assignment_function(random_medoids,df))
    print(f"\nDone with Medoid[2] movement check to Row no. {i} if the dataset\n")

  for j in range (0, tot_rows, 1) :
    X_1.append(list_int_1[j][6])

  list_int_2=[]
  X_2=[]

  for i1 in range (0, tot_rows, 1) :
    random_medoids=[Parent_Medoids[0],[df.iat[i1,0], df.iat[i1,1], df.iat[i1,2], df.iat[i1,3],
df.iat[i1,4]] ,Parent_Medoids[2]]
    list_int_2.append(assignment_function(random_medoids,df))
    print(f"\nDone with Medoid[1] movement check to Row no. {i1} if the dataset\n")

  for j1 in range (0, tot_rows, 1) :
    X_2.append(list_int_2[j1][6])

  list_int_3=[]
  X_3=[]

  for i2 in range (0, tot_rows, 1) :
    random_medoids=[[df.iat[i2,0], df.iat[i2,1], df.iat[i2,2], df.iat[i2,3], df.iat[i2,4]],
Parent_Medoids[1] ,Parent_Medoids[2]]
    list_int_3.append(assignment_function(random_medoids,df))
    print(f"\nDone with Medoid[0] movement check to Row no. {i2} if the dataset\n")

  for j2 in range (0, tot_rows, 1) :
    X_3.append(list_int_3[j2][6])


  min_SSE= min(X_1)
  min_SSE1= min(X_2)
  min_SSE2= min(X_3)

  min_SSE_ind=X_1.index(min_SSE)
  min_SSE_ind1=X_2.index(min_SSE1)
  min_SSE_ind2=X_3.index(min_SSE2)

  Ovr_min_SSE= min(min_SSE,min_SSE1,min_SSE2)

  if (Ovr_min_SSE == min_SSE) :
    if (min_SSE<SSE_tot) :
      cluster0=list_int_1[min_SSE_ind][0]
      cluster1=list_int_1[min_SSE_ind][1]
      cluster2=list_int_1[min_SSE_ind][2]
      SSE_C0=list_int_1[min_SSE_ind][3]
      SSE_C1=list_int_1[min_SSE_ind][4]
      SSE_C2=list_int_1[min_SSE_ind][5]
      SSE_Clusterings=list_int_1[min_SSE_ind][6]
      new_iteration_medoids = [Parent_Medoids[0],Parent_Medoids[1], [df.iat[min_SSE_ind,0],
df.iat[min_SSE_ind,1], df.iat[min_SSE_ind,2], df.iat[min_SSE_ind,3], df.iat[min_SSE_ind,4]]]
      Conv=0
```

```
    else :
      cluster0=C0
      cluster1=C1
      cluster2=C2
      SSE_C0=SSE0
      SSE_C1=SSE1
      SSE_C2=SSE2
      SSE_Clusterings=SSE_tot
      new_iteration_medoids = Parent_Medoids
      Conv=1
      print("\nConvergence Reached as no better medoid combination exists!\n")

  elif (Ovr_min_SSE == min_SSE1) :
    if (min_SSE1<SSE_tot) :
      cluster0=list_int_2[min_SSE_ind1][0]
      cluster1=list_int_2[min_SSE_ind1][1]
      cluster2=list_int_2[min_SSE_ind1][2]
      SSE_C0=list_int_2[min_SSE_ind1][3]
      SSE_C1=list_int_2[min_SSE_ind1][4]
      SSE_C2=list_int_2[min_SSE_ind1][5]
      SSE_Clusterings=list_int_2[min_SSE_ind1][6]
      new_iteration_medoids = [Parent_Medoids[0],[df.iat[min_SSE_ind1,0], df.iat[min_SSE_ind1,1],
df.iat[min_SSE_ind1,2], df.iat[min_SSE_ind1,3], df.iat[min_SSE_ind1,4]], Parent_Medoids[2]]
      Conv=0

    else :
      cluster0=C0
      cluster1=C1
      cluster2=C2
      SSE_C0=SSE0
      SSE_C1=SSE1
      SSE_C2=SSE2
      SSE_Clusterings=SSE_tot
      new_iteration_medoids = Parent_Medoids
      Conv=1
      print("\nConvergence Reached as no better medoid combination exists!\n")

  elif (Ovr_min_SSE == min_SSE2) :
    if (min_SSE2<SSE_tot) :
      cluster0=list_int_3[min_SSE_ind2][0]
      cluster1=list_int_3[min_SSE_ind2][1]
      cluster2=list_int_3[min_SSE_ind2][2]
      SSE_C0=list_int_3[min_SSE_ind2][3]
      SSE_C1=list_int_3[min_SSE_ind2][4]
      SSE_C2=list_int_3[min_SSE_ind2][5]
      SSE_Clusterings=list_int_3[min_SSE_ind2][6]
      new_iteration_medoids = [[df.iat[min_SSE_ind2,0], df.iat[min_SSE_ind2,1],
df.iat[min_SSE_ind2,2], df.iat[min_SSE_ind2,3], df.iat[min_SSE_ind2,4]],Parent_Medoids[1],
Parent_Medoids[2]]
      Conv=0

    else :
      cluster0=C0
      cluster1=C1
      cluster2=C2
      SSE_C0=SSE0
```

```
      SSE_C1=SSE1
      SSE_C2=SSE2
      SSE_Clusterings=SSE_tot
      new_iteration_medoids = Parent_Medoids
      Conv=1
      print("\nConvergence Reached as no better medoid combination exists!\n")



  return  new_iteration_medoids, cluster0, cluster1, cluster2, SSE_C0, SSE_C1, SSE_C2, 
SSE_Clusterings, Conv


#****************************************************************************************************
****************************************************************************************************
***************************
#****************************************************************************************************
****************************************************************************************************
***************************

def kmedoids_iris(ini_medoids,dataset) :
  K=3
  Last_iteration_medoids = ini_medoids
  Iterations=0
  New_iteration_medoids=[[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]

  while 1:

    Iterations = Iterations+1
    print(f"\n\nIteration Number : {Iterations}\n\n")
    A1 = assignment_function(Last_iteration_medoids,dataset)
    A = A1[0]
    B = A1[1]
    C = A1[2]
    D = A1[3]
    E = A1[4]
    F = A1[5]
    G = A1[6]
    H = A1[7]
    print("Assignment is Done")

    B1=updation_function(A,B,C,D,E,F,G,H)
    New_iteration_medoids = B1[0]

    print(New_iteration_medoids)

    print("Updation is Done")
    C_0 = B1[1]
    C_1 = B1[2]
    C_2 = B1[3]
    len_C_0 = len(C_0.index)
    len_C_1 = len(C_1.index)
    len_C_2 = len(C_2.index)
    SSE_C_0 = B1[4]
    SSE_C_1 = B1[5]
```

```
    SSE_C_2 = B1[6]
    SSE_Total = SSE_C_0 + SSE_C_1 + SSE_C_2
    Convergence = B1[8]

    if (New_iteration_medoids==Last_iteration_medoids or Convergence==1) :


print("\n\n****************************************************************************************************************************************************************************************************************************")

print("\n****************************************************************************************************************************************************************************************************************************")
      print("\n\n")
      print(f"Iteration No. : {Iterations}\n\n")
      print(f"Total Number of Clusters : {K}")
      print("\n\n")
      print(f"Final Medoids : {New_iteration_medoids}")
      print("\n\n")
      print(f"Points in Cluster 0 : {len_C_0}")
      print("\n\n")
      print(f"SSE of Cluster 0 : {SSE_C_0}")
      print("\n\n")
      print(f"Points in Cluster 1 : {len_C_1}")
      print("\n\n")
      print(f"SSE of Cluster 1 : {SSE_C_1}")
      print("\n\n")
      print(f"Points in Cluster 2 : {len_C_2}")
      print("\n\n")
      print(f"SSE of Cluster 2 : {SSE_C_2}")
      print("\n\n")
      print(f"Total SSE : {SSE_Total}")


      sepallen_C_0=[]
      sepallen_C_1=[]
      sepallen_C_2=[]

      sepalwid_C_0=[]
      sepalwid_C_1=[]
      sepalwid_C_2=[]

      petallen_C_0=[]
      petallen_C_1=[]
      petallen_C_2=[]

      petalwid_C_0=[]
      petalwid_C_1=[]
      petalwid_C_2=[]

      sepallen_X_petallen_C_0=[]
      sepallen_X_petallen_C_1=[]
      sepallen_X_petallen_C_2=[]

      sepalwid_X_petalwid_C_0=[]
```

```
    sepalwid_X_petalwid_C_1=[]
    sepalwid_X_petalwid_C_2=[]

    for z in range (0,len(C_0.index),1) :
      sepallen_C_0.append(C_0.iat[z,0])
      sepalwid_C_0.append(C_0.iat[z,1])
      petallen_C_0.append(C_0.iat[z,2])
      petalwid_C_0.append(C_0.iat[z,3])
      sepallen_X_petallen_C_0.append(((7*C_0.iat[z,0])) + (3*C_0.iat[z,2]))
      sepalwid_X_petalwid_C_0.append((11*C_0.iat[z,1]) + (C_0.iat[z,3]))

    for y in range (0,len(C_1.index),1) :
      sepallen_C_1.append(C_1.iat[y,0])
      sepalwid_C_1.append(C_1.iat[y,1])
      petallen_C_1.append(C_1.iat[y,2])
      petalwid_C_1.append(C_1.iat[y,3])
      sepallen_X_petallen_C_1.append(((7*C_1.iat[y,0])) + (3*C_1.iat[y,2]))
      sepalwid_X_petalwid_C_1.append((11*C_1.iat[y,1]) + (C_1.iat[y,3]))

    for x in range (0,len(C_2.index),1) :
      sepallen_C_2.append(C_2.iat[x,0])
      sepalwid_C_2.append(C_2.iat[x,1])
      petallen_C_2.append(C_2.iat[x,2])
      petalwid_C_2.append(C_2.iat[x,3])
      sepallen_X_petallen_C_2.append(((7*C_2.iat[x,0])) + (3*C_2.iat[x,2]))
      sepalwid_X_petalwid_C_2.append((11*C_2.iat[x,1]) + (C_2.iat[x,3]))

    plt.scatter(sepallen_C_0,sepalwid_C_0, c='green')
    plt.scatter(sepallen_C_1,sepalwid_C_1, c='blue')
    plt.scatter(sepallen_C_2,sepalwid_C_2, c='red')
    plt.xlabel('sepal length in cm')
    plt.ylabel('sepal width in cm')
    plt.title('Clusterings')
    plt.show()

    plt.scatter(petallen_C_0,petalwid_C_0, c='green')
    plt.scatter(petallen_C_1,petalwid_C_1, c='blue')
    plt.scatter(petallen_C_2,petalwid_C_2, c='red')
    plt.xlabel('petal length in cm')
    plt.ylabel('petal width in cm')
    plt.title('Clusterings')
    plt.show()

    plt.scatter(sepallen_C_0,petallen_C_0, c='green')
    plt.scatter(sepallen_C_1,petallen_C_1, c='blue')
    plt.scatter(sepallen_C_2,petallen_C_2, c='red')
    plt.xlabel('sepal length in cm')
    plt.ylabel('petal length in cm')
    plt.title('Clusterings')
    plt.show()

    plt.scatter(sepallen_C_0,petalwid_C_0, c='green')
    plt.scatter(sepallen_C_1,petalwid_C_1, c='blue')
    plt.scatter(sepallen_C_2,petalwid_C_2, c='red')
    plt.xlabel('sepal length in cm')
    plt.ylabel('petal width in cm')
```

```
      plt.title('Clusterings')
      plt.show()

      plt.scatter(sepalwid_C_0,petalwid_C_0, c='green')
      plt.scatter(sepalwid_C_1,petalwid_C_1, c='blue')
      plt.scatter(sepalwid_C_2,petalwid_C_2, c='red')
      plt.xlabel('sepal width in cm')
      plt.ylabel('petal width in cm')
      plt.title('Clusterings')
      plt.show()


      plt.scatter(sepalwid_C_0,petallen_C_0, c='green')
      plt.scatter(sepalwid_C_1,petallen_C_1, c='blue')
      plt.scatter(sepalwid_C_2,petallen_C_2, c='red')
      plt.xlabel('sepal width in cm')
      plt.ylabel('petal length in cm')
      plt.title('Clusterings')
      plt.show()

      plt.scatter(sepallen_X_petallen_C_0,sepalwid_X_petalwid_C_0, c='green')
      plt.scatter(sepallen_X_petallen_C_1,sepalwid_X_petalwid_C_1, c='blue')
      plt.scatter(sepallen_X_petallen_C_2,sepalwid_X_petalwid_C_2, c='red')
      plt.xlabel('Linear Sum  of Sepal & Petal lengths in cm')
      plt.ylabel('Linear Sum  of Sepal & Petal widths in cm')
      plt.title('Clusterings')
      plt.show()

      break

    elif (New_iteration_medoids!=Last_iteration_medoids) :

print("\n\n********************************************************************************************
******************************************************************************************************
********************************")
      Last_iteration_medoids = New_iteration_medoids
      print(f"\n\nChecking for Convergence in K-Medoids Algorithm in Iteration number :
{Iterations}\n\n")


#*****************************************************************************************************
******************************************************************************************************
****************************
#*****************************************************************************************************
******************************************************************************************************
****************************
medoid_1 = [[5.1,3.3,1.7,0.5,0], [6.1,2.8,4.7,1.2,1], [6.3,2.7,4.9,1.8,2]]
#2h56mins, [50,47,53], SSE : 128.13

kmedoids_iris(medoid_1,df)


#*****************************************************************************************************
******************************************************************************************************
****************************
```