Code:

#bisecting_kmeans.py

```
import random
import math
import numpy
import k_means_algo_new as km
import graphToolKit as gtk

def findSquaredDistance(obj1, obj2):
    distance = 0
    for i in range(len(obj1)):
        distance += (obj1[i] - obj2[i])**2
    return distance

def findSSE(clusterList, clusterListLen):
    sseList = findICD(clusterList, clusterListLen)
    return sum(sseList)

def findICD(clusterList, clusterListLen):
    sseList = []
    for i in range(clusterListLen):
        sse = 0
        centroid = clusterList["cent" + str(i + 1)]
        for obj in clusterList["cluster" + str(i + 1)]:
            sse += findSquaredDistance(obj, centroid)
        sseList.append(sse)
    return sseList

# taking input from file
dataSet = []
dataFile = open("iris.data", "r")
for line in dataFile:
    obj = []
    x = line.strip().split(",")
    for i in range(4):
        obj.append((float)(x[i]))
    dataSet.append(obj)
random.shuffle(dataSet)

# gtk.plot2DGraphOriginal(dataSet)

finalClusters = {}
minimalSSE = 0
```

```
sseValues = []

for i in range(500):
   clusterList = {}
   clusterListLen = 1
   clusterList["cluster1"] = dataSet

   while True:
       # cluster = random.randrange(clusterListLen) + 1
       if clusterListLen == 1:
           cluster = 1
       else:
           sseList = findICD(clusterList, 2)
           cluster = sseList.index(max(sseList)) + 1
       bisectingCluster = clusterList["cluster" + str(cluster)]

       # returns a dictionary with best clustering sln and their centroids
       # {
       # "cent1" : value1,
       # "cent2" : value2,
       # "cluster1": cluster1,
       # "cluster2": cluster2
       # }

       bisectedClusters = km.kMeansAlgo(bisectingCluster)

       # update the clusterList
       clusterListLen += 1
       clusterList["cluster" + str(cluster)] = bisectedClusters["cluster1"]
       clusterList["cent" + str(cluster)] = bisectedClusters["cent1"]
       clusterList["cluster" + str(clusterListLen)] = bisectedClusters["cluster2"]
       clusterList["cent" + str(clusterListLen)] = list(bisectedClusters["cent2"])

       if clusterListLen == 3:
           break

   newSSE = findSSE(clusterList, clusterListLen)

   if newSSE < minimalSSE or i == 0:
       minimalSSE = newSSE
       sseValues.append(newSSE)

       finalClusters["cluster1"] = clusterList["cluster1"]
       finalClusters["cluster2"] = clusterList["cluster2"]
```

```
        finalClusters["cluster3"] = clusterList["cluster3"]
    else:
        sseValues.append(sseValues[-1])

print("minimalSSE in bisecting k means= " + str(minimalSSE))
for cluster in finalClusters:
    print(cluster)
    print(finalClusters[cluster])

gtk.plotSSEGraphToCompare(sseValues, 500)
gtk.plot4DGraph(finalClusters)
gtk.plot3DGraph(finalClusters)
gtk.plot2DGraph(finalClusters)
gtk.plotSSEGraph(sseValues, 500)
```

#graph_Tool_Kit.py

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import k_means_algo_mod as kmm
import numpy

def plot2DGraphOriginal(dataSet):
    cluster1 = []
    cluster2 = []
    cluster3 = []
    i = 0
    for obj in dataSet:
        if i < 50:
            cluster1.append(obj)
        elif i < 100:
            cluster2.append(obj)
        else:
            cluster3.append(obj)

        i += 1

    newCluster = {}
    newCluster["cluster1"] = cluster1
    newCluster["cluster2"] = cluster2
    newCluster["cluster3"] = cluster3

sseList = []
    for i in range(3):
        sse = 0
        cent = kmm.findMean(newCluster["cluster" + str(i + 1)])
        for obj in newCluster["cluster" + str(i + 1)]:
            sse += kmm.findSquaredDistance(obj, cent)
        sseList.append(sse)
    print("original dataset centroids = " + str(sseList))
    print("original clusters' SSE = " + str(sum(sseList)))

    plot2DGraph(newCluster)

def plot4DGraph(clusters):

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
```

```
    iter = 0
    for cluster in clusters:
        u_val = [obj[0] for obj in clusters[cluster]]
        v_val = [obj[1] for obj in clusters[cluster]]
        w_val = [obj[2] for obj in clusters[cluster]]
        x_val = [obj[3] for obj in clusters[cluster]]

        if iter == 0:
            img1 = ax.scatter(u_val, v_val, w_val, s = 75, c = x_val, cmap =
plt.winter(), label = 'cluster1')
            cbar = fig.colorbar(img1, shrink = 0.5, aspect = 10)
        elif iter == 1:
            img2 = ax.scatter(u_val, v_val, w_val, s = 75, c = x_val, cmap =
plt.spring(), label = 'cluster2')
            cbar = fig.colorbar(img2, shrink = 0.5, aspect = 10)
        else:
            img3 = ax.scatter(u_val, v_val, w_val, s = 75, c = x_val, cmap =
plt.gray(), label = 'cluster3')
            cbar = fig.colorbar(img3, shrink = 0.5, aspect = 10)

        iter += 1
        cbar.ax.get_yaxis().labelpad = 15
        cbar.ax.set_ylabel('petal width in cm')
        cbar.ax.get_xaxis().labelpad = 15
        cbar.ax.set_xlabel('cluster' + str(iter))

    ax.set_xlabel('sepal length in cm', rotation=150)
    ax.set_ylabel('sepal width in cm')
    ax.set_zlabel(r'petal length in cm', rotation=60)


    plt.title("4D representation of clustering solution")
    plt.show()

def plot3DGraph(clusters):

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    colorArray = ['red', 'green', 'blue']
    iter = 0

    for cluster in clusters:
        u_val = [obj[0] for obj in clusters[cluster]]
        v_val = [obj[1] for obj in clusters[cluster]]
```

```
        w_val = [obj[2] for obj in clusters[cluster]]

        ax.scatter(u_val, v_val, w_val, s = 75, c = colorArray[iter], label =
'cluster' + str(iter + 1))
        iter += 1

    plt.legend()
    ax.set_xlabel('sepal length in cm', fontsize=13, rotation=150)
    ax.set_ylabel('sepal width in cm', fontsize=13)
    ax.set_zlabel(r'petal length in cm', fontsize=13, rotation=60)
    plt.title("3D representation of clustering solution")
    plt.show()

def plotSSEGraph(sseValues, iter):

    # change wrt number of iterations
    x_val = numpy.arange(1, iter + 1 ,1)
    y_val = sseValues
    plt.plot(x_val, y_val)
    plt.scatter(x_val, y_val, c = "red", marker= '+', label = "sse value")

    plt.xlabel("iteration")
    plt.ylabel("SSE values")
    plt.title("iteration vs SSE values")
    plt.grid()
    plt.legend()
    plt.show()

def plotSSEGraphToCompare(sseValues, iter):
    newSSEvalues = kmm.NormalKmeans()

    x_val = numpy.arange(1, iter + 1 ,1)
    y_val = sseValues
    plt.plot(x_val, y_val)
    plt.scatter(x_val, y_val, c = "red", marker= '+', label = "Bisecting Kmeans")

    y_val = newSSEvalues
    plt.plot(x_val, y_val)
    plt.scatter(x_val, y_val, c = "green", marker= '*', label = "Kmeans")

    plt.xlabel("iteration")
    plt.ylabel("SSE values")
    plt.title("iteration vs SSE values: comparison b/w K Means and Bisecting K
means")
```

```
    plt.grid()
    plt.legend()
    plt.show()

def plot2DGraph(clusters):

    colorArray = ['red', 'green', 'blue']
    attributes = ["sepal length", "sepal width", "petal length", "petal width"]
    for i in range(0,3,2):
        iter = 0
        for cluster in clusters:
            u_val = [obj[0 + i] for obj in clusters[cluster]]
            v_val = [obj[1 + i] for obj in clusters[cluster]]

            plt.scatter(u_val, v_val, s = 50, c = colorArray[iter], label =
"cluster" + str(iter + 1))
            iter += 1

        # plt.grid()
        plt.xlabel(attributes[0 + i] + "(cm)", fontsize = 15)
        plt.ylabel(attributes[1 + i] + "(cm)", fontsize = 15)
        plt.title(attributes[0 + i] + " vs " + attributes[1 + i] + " of clusters",
fontsize = 20)
        plt.show()
```

#k_means_algo_new.py

```
import math
import numpy
import random

def findDistance(obj1, obj2):
    distance = 0
    for i in range(len(obj1)):
        distance += (obj1[i] - obj2[i])**2
    return math.sqrt(distance)

def findSquaredDistance(obj1, obj2):
    distance = 0
    for i in range(len(obj1)):
        distance += (obj1[i] - obj2[i])**2
    return distance

def findCluster(obj1, cent1, cent2):
    distances = []
    distances.append(findDistance(obj1, cent1))
    distances.append(findDistance(obj1, cent2))
    return distances.index(min(distances)) + 1

def findMean(cluster):
    uval = wval = xval = yval = 0
    for obj in cluster:
        uval += obj[0]
        wval += obj[1]
        xval += obj[2]
        yval += obj[3]
    size = len(cluster)
    return [(uval/size), (wval/size), (xval/size), (yval/size)]

def findSSE(centroids, clusters):
    sse = 0
    for i in range(2):
        for obj in clusters["cluster" + str(i + 1)]:
            sse += findSquaredDistance(obj, centroids[i])
    return sse
```

```
def kMeansAlgo(bisectingCluster):
    finalClusters = {}
    minimalSSE = 0
    sseValues = []

    for i in range(30):
        # initialize centroid values for bisecting without repetition
        while True:
            cent =  numpy.array(random.sample(bisectingCluster, 2))
            if not numpy.array_equal(cent[0], cent[1]):
                break

        cluster1 = []
        cluster2 = []

        while True:
            cluster1.clear()
            cluster2.clear()
            for obj in bisectingCluster:
                cluster = findCluster(obj, cent[0], cent[1])
                if cluster == 1:
                    cluster1.append(obj)
                else:
                    cluster2.append(obj)

            newCent = numpy.array([findMean(cluster1), findMean(cluster2)])

            compare = cent == newCent

            if compare.all() :
                break
            else:
                   cent = numpy.delete(cent,[0,1],0)
                cent = newCent

        # print("iteration: " + str(i))
        clusters = {}
        clusters["cluster1"] = cluster1
        clusters["cluster2"] = cluster2

        newSSE = findSSE(cent, clusters)

        if newSSE < minimalSSE or i == 0:
            minimalSSE = newSSE
```

```
            sseValues.append(newSSE)

            finalClusters.clear()
            finalClusters["cluster1"] = clusters["cluster1"]
            finalClusters["cluster2"] = clusters["cluster2"]
            finalClusters["cent1"] = list(cent[0])
            finalClusters["cent2"] = list(cent[1])
        else:
            sseValues.append(sseValues[-1])

    return finalClusters
```

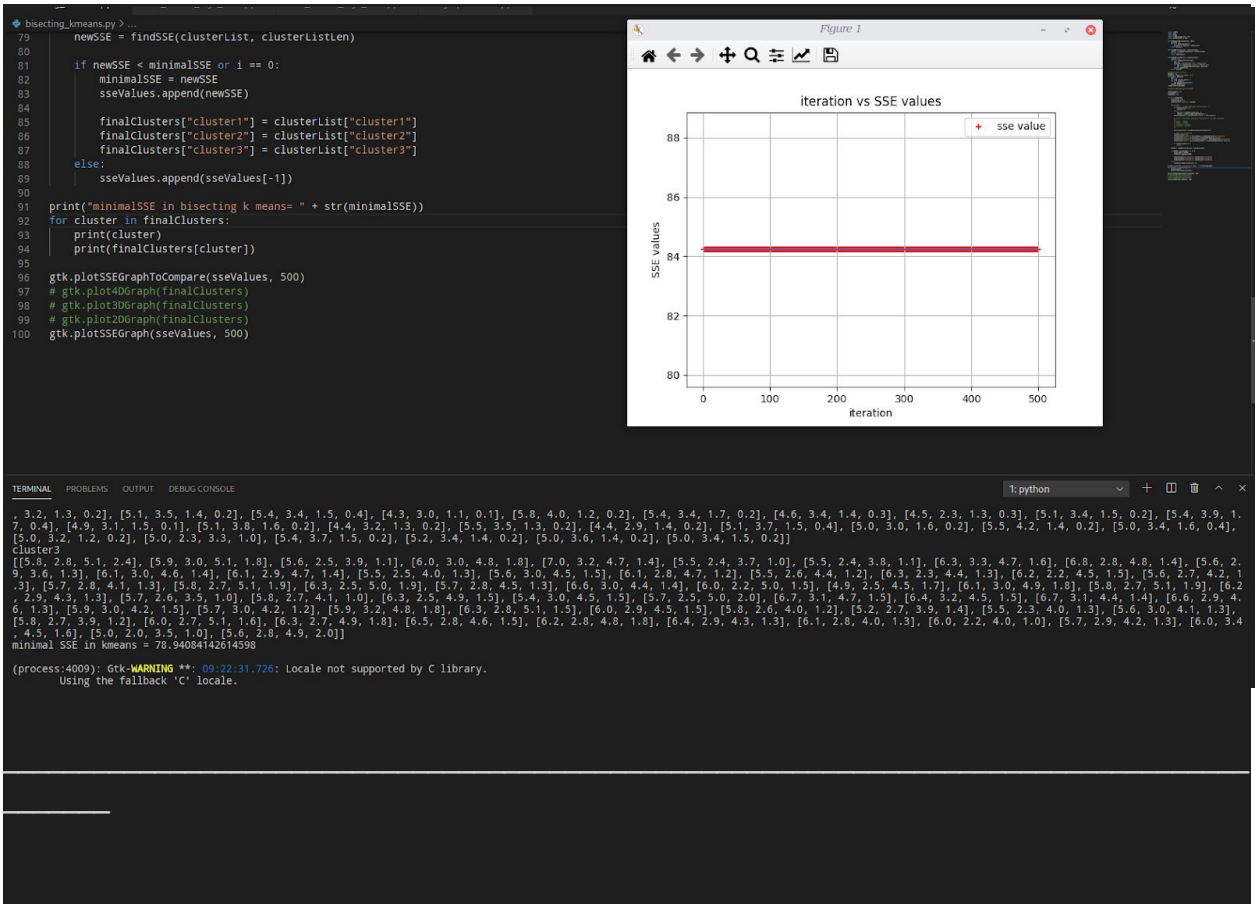
#k_means_algo_mod.py

```
import math
import numpy
import random
import graphToolKit as gtk

def findDistance(obj1, obj2):
    distance = 0
    for i in range(len(obj1)):
        distance += (obj1[i] - obj2[i])**2
    return math.sqrt(distance)

def findSquaredDistance(obj1, obj2):
    distance = 0
    for i in range(len(obj1)):
        distance += (obj1[i] - obj2[i])**2
    return distance

def findCluster(obj1, cent1, cent2, cent3):
    distances = []
    distances.append(findDistance(obj1, cent1))
    distances.append(findDistance(obj1, cent2))
    distances.append(findDistance(obj1, cent3))
    return distances.index(min(distances)) + 1

def findMean(cluster):
    uval = wval = xval = yval = 0
    for obj in cluster:
        uval += obj[0]
        wval += obj[1]
        xval += obj[2]
        yval += obj[3]
    size = len(cluster)
    return [(uval/size), (wval/size), (xval/size), (yval/size)]

def findSSE(centroids, clusters):
    sse = 0
    for i in range(3):
        for obj in clusters["cluster" + str(i + 1)]:
            sse += findSquaredDistance(obj, centroids[i])
```

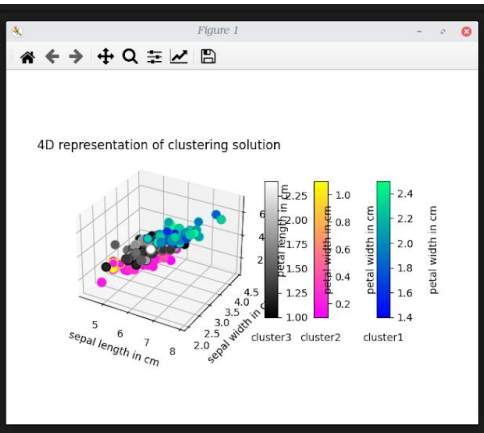
```
    return sse

def NormalKmeans():

    # taking input from file
    dataSet = []
    dataFile = open("iris.data", "r")
    for line in dataFile:
        obj = []
        x = line.strip().split(",")
        for i in range(4):
            obj.append((float)(x[i]))
        dataSet.append(obj)

    random.shuffle(dataSet)

    finalClusters = {}
    minimalSSE = 0
    sseValues = []

    for i in range(500):
        # initialize centroid values
        while True:
            cent =  numpy.array(random.sample(dataSet, 3))
            if not (numpy.array_equal(cent[0], cent[1]) or
numpy.array_equal(cent[0], cent[2]) or numpy.array_equal(cent[1], cent[2])):
                break
        cluster1 = []
        cluster2 = []
        cluster3 = []

        while True:
            cluster1.clear()
            cluster2.clear()
            cluster3.clear()
            for obj in dataSet:
                cluster = findCluster(obj, cent[0], cent[1], cent[2])
                if cluster == 1:
                    cluster1.append(obj)
                elif cluster == 2:
                    cluster2.append(obj)
                else:
                    cluster3.append(obj)
```

```
            newCent = numpy.array([findMean(cluster1), findMean(cluster2),
findMean(cluster3)])

            compare = cent == newCent

            if compare.all() :
                break
            else:

                cent = numpy.delete(cent,[0,1,2],0)
                cent = newCent

        clusters = {}
        clusters["cluster1"] = cluster1
        clusters["cluster2"] = cluster2
        clusters["cluster3"] = cluster3

        newSSE = findSSE(cent, clusters)
        # sseValues.append(newSSE)

        if newSSE < minimalSSE or i == 0:
            minimalSSE = newSSE
            sseValues.append(newSSE)
            finalClusters.clear()
            for cluster in clusters:
                finalClusters.update({cluster : clusters[cluster]})
        else:
            sseValues.append(sseValues[-1])

    print("minimal SSE in kmeans = " + str(minimalSSE))

    return sseValues
```

Screenshots:



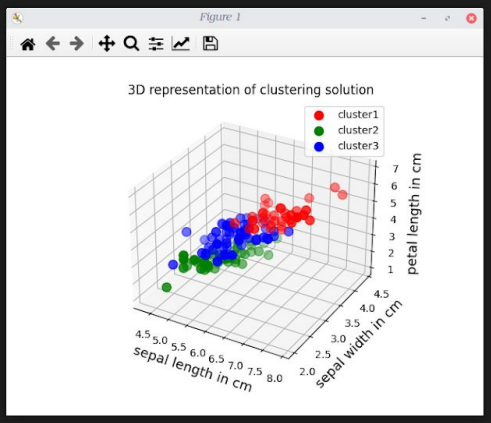
```
        sseValues.append(newSSE)

        finalClusters["cluster1"] = clusterList["cluster1"]
        finalClusters["cluster2"] = clusterList["cluster2"]
        finalClusters["cluster3"] = clusterList["cluster3"]
    else:
        sseValues.append(sseValues[-1])

print("minimalSSE in bisecting k means= " + str(minimalSSE))
for cluster in finalClusters:
    print(cluster)
    print(finalClusters[cluster])

gtk.plotSSEGraphToCompare(sseValues)
gtk.plot4DGraph(finalClusters)
gtk.plot3DGraph(finalClusters)
gtk.plot2DGraph(finalClusters)
gtk.plotSSEGraph(sseValues)
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: python

```
, 3.4, 1.6, 0.2], [5.0, 3.0, 1.6, 0.2], [5.1, 3.8, 1.6, 0.2], [4.9, 3.1, 1.5, 0.1], [5.1, 3.8, 1.9, 0.4], [5.4, 3.9, 1.3, 0.4], [5.0, 2.3, 3.3, 1.0], [5.1, 3.5, 1.4, 0.3], [4.9, 3.1, 1.5, 0.1], [4.4, 2.9, 1.
4, 0.2], [5.1, 3.7, 1.5, 0.4], [5.5, 4.2, 1.4, 0.2], [4.4, 3.2, 1.3, 0.2], [5.1, 2.5, 3.0, 1.1], [4.8, 3.4, 1.9, 0.2], [4.9, 3.1, 1.5, 0.1], [4.4, 3.0, 1.3, 0.2], [4.3, 3.0, 1.1, 0.1], [5.2, 3.5, 1.5, 0.2],
[4.6, 3.6, 1.0, 0.2], [5.0, 3.5, 1.3, 0.3], [5.1, 3.4, 1.5, 0.2], [5.0, 3.3, 1.4, 0.2], [5.3, 3.7, 1.5, 0.2], [4.6, 3.2, 1.4, 0.2]]
cluster3
[[4.9, 2.5, 4.5, 1.7], [5.8, 2.7, 4.1, 1.0], [5.6, 3.0, 4.1, 1.3], [5.0, 2.0, 3.5, 1.0], [6.5, 2.8, 4.6, 1.5], [6.1, 2.9, 4.7, 1.4], [6.3, 2.5, 4.9, 1.5], [6.2, 2.2, 4.5, 1.5], [6.0, 3.4, 4.5, 1.6], [5.6, 2.
9, 3.6, 1.3], [5.5, 2.5, 4.0, 1.3], [6.1, 2.8, 4.7, 1.2], [6.3, 2.5, 5.0, 1.9], [6.1, 3.0, 4.9, 1.8], [6.6, 3.0, 4.4, 1.4], [6.0, 2.2, 5.0, 1.5], [5.6, 3.0, 4.5, 1.5], [5.6, 2.5, 3.9, 1.1], [6.7, 3.1, 4.4, 1
.4], [5.5, 2.4, 3.8, 1.1], [6.0, 2.7, 5.1, 1.6], [5.7, 2.8, 4.1, 1.3], [5.8, 2.6, 4.0, 1.2], [5.5, 2.4, 3.7, 1.0], [6.6, 2.9, 4.6, 1.3], [6.3, 2.3, 4.4, 1.3], [6.2, 2.9, 4.3, 1.3], [5.8, 2.7, 5.1, 1.9], [5.7
, 2.9, 4.2, 1.3], [6.4, 3.2, 4.5, 1.5], [6.0, 3.0, 4.8, 1.8], [5.7, 3.0, 4.2, 1.2], [5.9, 3.2, 4.8, 1.8], [5.6, 2.8, 4.9, 2.0], [5.6, 2.7, 4.2, 1.3], [5.4, 3.0, 4.5, 1.5], [6.0, 2.2, 4.0, 1.0], [5.7, 2.6, 3.
5, 1.0], [7.0, 3.2, 4.7, 1.4], [6.0, 2.9, 4.5, 1.5], [6.3, 2.8, 5.1, 1.5], [5.9, 3.0, 4.2, 1.5], [5.2, 2.7, 3.9, 1.4], [6.2, 2.8, 4.8, 1.8], [5.7, 2.5, 5.0, 2.0], [6.1, 2.8, 4.0, 1.3], [5.8, 2.7, 3.9, 1.2],
[5.5, 2.3, 4.0, 1.3], [6.3, 2.7, 4.9, 1.8], [5.7, 2.8, 4.5, 1.3], [5.9, 3.0, 5.1, 1.8], [6.8, 2.8, 4.8, 1.4], [6.4, 2.9, 4.3, 1.3], [6.1, 3.0, 4.6, 1.4], [5.5, 2.6, 4.4, 1.2], [5.8, 2.8, 5.1, 2.4], [5.8, 2.7
, 5.1, 1.9], [6.7, 3.1, 4.7, 1.5], [6.3, 3.3, 4.7, 1.6]]
minimal SSE in kmeans = 78.940841426146

(process:8853): Gtk-WARNING **: 16:49:07.660: Locale not supported by C library.
        Using the fallback 'C' locale.

(python:8853): Gtk-WARNING **: 16:49:07.677: Theme parsing error: gtk.css:3611:22: 'none' is not a valid color name
gt.qpa.xcb: QXcbConnection: XCB error: 5 (BadAtom), sequence: 685, resource id: 0, major code: 20 (GetProperty), minor code: 0
```

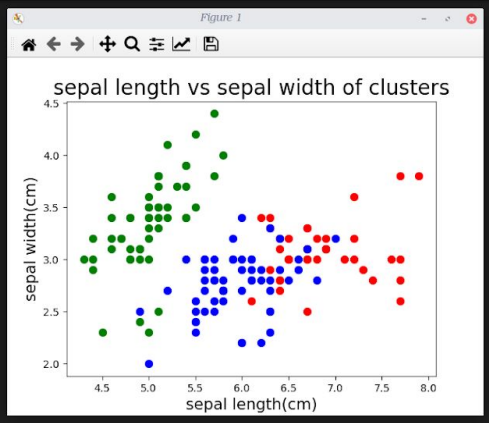
```
        sseValues.append(newSSE)

        finalClusters["cluster1"] = clusterList["cluster1"]
        finalClusters["cluster2"] = clusterList["cluster2"]
        finalClusters["cluster3"] = clusterList["cluster3"]
    else:
        sseValues.append(sseValues[-1])

print("minimalSSE in bisecting k means= " + str(minimalSSE))
for cluster in finalClusters:
    print(cluster)
    print(finalClusters[cluster])

gtk.plotSSEGraphToCompare(sseValues)
gtk.plot4DGraph(finalClusters)
gtk.plot3DGraph(finalClusters)
gtk.plot2DGraph(finalClusters)
gtk.plotSSEGraph(sseValues)
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: python

```
, 3.4, 1.6, 0.2], [5.0, 3.0, 1.6, 0.2], [5.1, 3.8, 1.6, 0.2], [4.9, 3.1, 1.5, 0.1], [5.1, 3.8, 1.9, 0.4], [5.4, 3.9, 1.3, 0.4], [5.0, 2.3, 3.3, 1.0], [5.1, 3.5, 1.4, 0.3], [4.9, 3.1, 1.5, 0.1], [4.4, 2.9, 1.
4, 0.2], [5.1, 3.7, 1.5, 0.4], [5.5, 4.2, 1.4, 0.2], [4.4, 3.2, 1.3, 0.2], [5.1, 2.5, 3.0, 1.1], [4.8, 3.4, 1.9, 0.2], [4.9, 3.1, 1.5, 0.1], [4.4, 3.0, 1.3, 0.2], [4.3, 3.0, 1.1, 0.1], [5.2, 3.5, 1.5, 0.2],
[4.6, 3.6, 1.0, 0.2], [5.0, 3.5, 1.3, 0.3], [5.1, 3.4, 1.5, 0.2], [5.0, 3.3, 1.4, 0.2], [5.3, 3.7, 1.5, 0.2], [4.6, 3.2, 1.4, 0.2]]
cluster3
[[4.9, 2.5, 4.5, 1.7], [5.8, 2.7, 4.1, 1.0], [5.6, 3.0, 4.1, 1.3], [5.0, 2.0, 3.5, 1.0], [6.5, 2.8, 4.6, 1.5], [6.1, 2.9, 4.7, 1.4], [6.3, 2.5, 4.9, 1.5], [6.2, 2.2, 4.5, 1.5], [6.0, 3.4, 4.5, 1.6], [5.6, 2.
9, 3.6, 1.3], [5.5, 2.5, 4.0, 1.3], [6.1, 2.8, 4.7, 1.2], [6.3, 2.5, 5.0, 1.9], [6.1, 3.0, 4.9, 1.8], [6.6, 3.0, 4.4, 1.4], [6.0, 2.2, 5.0, 1.5], [5.6, 3.0, 4.5, 1.5], [5.6, 2.5, 3.9, 1.1], [6.7, 3.1, 4.4, 1
.4], [5.5, 2.4, 3.8, 1.1], [6.0, 2.7, 5.1, 1.6], [5.7, 2.8, 4.1, 1.3], [5.8, 2.6, 4.0, 1.2], [5.5, 2.4, 3.7, 1.0], [6.6, 2.9, 4.6, 1.3], [6.3, 2.3, 4.4, 1.3], [6.2, 2.9, 4.3, 1.3], [5.8, 2.7, 5.1, 1.9], [5.7
, 2.9, 4.2, 1.3], [6.4, 3.2, 4.5, 1.5], [6.0, 3.0, 4.8, 1.8], [5.7, 3.0, 4.2, 1.2], [5.9, 3.2, 4.8, 1.8], [5.6, 2.8, 4.9, 2.0], [5.6, 2.7, 4.2, 1.3], [5.4, 3.0, 4.5, 1.5], [6.0, 2.2, 4.0, 1.0], [5.7, 2.6, 3.
5, 1.0], [7.0, 3.2, 4.7, 1.4], [6.0, 2.9, 4.5, 1.5], [6.3, 2.8, 5.1, 1.5], [5.9, 3.0, 4.2, 1.5], [5.2, 2.7, 3.9, 1.4], [6.2, 2.8, 4.8, 1.8], [5.7, 2.5, 5.0, 2.0], [6.1, 2.8, 4.0, 1.3], [5.8, 2.7, 3.9, 1.2],
[5.5, 2.3, 4.0, 1.3], [6.3, 2.7, 4.9, 1.8], [5.7, 2.8, 4.5, 1.3], [5.9, 3.0, 5.1, 1.8], [6.8, 2.8, 4.8, 1.4], [6.4, 2.9, 4.3, 1.3], [6.1, 3.0, 4.6, 1.4], [5.5, 2.6, 4.4, 1.2], [5.8, 2.8, 5.1, 2.4], [5.8, 2.7
, 5.1, 1.9], [6.7, 3.1, 4.7, 1.5], [6.3, 3.3, 4.7, 1.6]]
minimal SSE in kmeans = 78.940841426146

(process:8853): Gtk-WARNING **: 16:49:07.660: Locale not supported by C library.
        Using the fallback 'C' locale.

(python:8853): Gtk-WARNING **: 16:49:07.677: Theme parsing error: gtk.css:3611:22: 'none' is not a valid color name
gt.qpa.xcb: QXcbConnection: XCB error: 5 (BadAtom), sequence: 685, resource id: 0, major code: 20 (GetProperty), minor code: 0
```

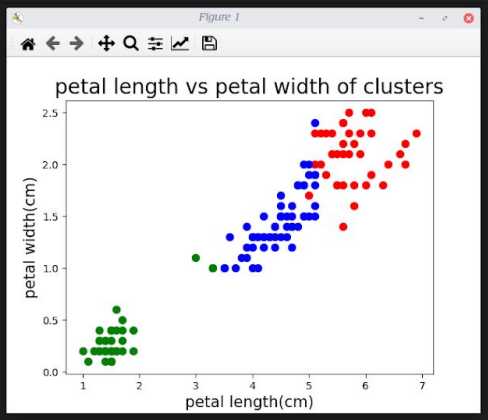
```
            sseValues.append(newSSE)

            finalClusters["cluster1"] = clusterList["cluster1"]
            finalClusters["cluster2"] = clusterList["cluster2"]
            finalClusters["cluster3"] = clusterList["cluster3"]
        else:
            sseValues.append(sseValues[-1])

    print("minimalSSE in bisecting k means= " + str(minimalSSE))
    for cluster in finalClusters:
        print(cluster)
        print(finalClusters[cluster])

    gtk.plotSSEGraphToCompare(sseValues)
    gtk.plot4DGraph(finalClusters)
    gtk.plot3DGraph(finalClusters)
    gtk.plot2DGraph(finalClusters)
    gtk.plotSSEGraph(sseValues)
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: python

```
, 3.4, 1.6, 0.2], [5.0, 3.0, 1.6, 0.2], [5.1, 3.8, 1.6, 0.2], [4.9, 3.1, 1.5, 0.1], [5.1, 3.8, 1.9, 0.4], [5.4, 3.9, 1.3, 0.4], [5.0, 2.3, 3.3, 1.0], [5.1, 3.5, 1.4, 0.3], [4.9, 3.1, 1.5, 0.1], [4.4, 2.9, 1.
4, 0.2], [5.1, 3.7, 1.5, 0.4], [5.5, 4.2, 1.4, 0.2], [4.4, 3.2, 1.3, 0.2], [5.1, 2.5, 3.0, 1.1], [4.8, 3.4, 1.9, 0.2], [4.9, 3.1, 1.5, 0.1], [4.4, 3.0, 1.3, 0.2], [4.3, 3.0, 1.1, 0.1], [5.2, 3.5, 1.5, 0.2],
[4.6, 3.6, 1.0, 0.2], [5.0, 3.5, 1.3, 0.3], [5.1, 3.4, 1.5, 0.2], [5.0, 3.3, 1.4, 0.2], [5.3, 3.7, 1.5, 0.2], [4.6, 3.2, 1.4, 0.2]]
cluster3
[[4.9, 2.5, 4.5, 1.7], [5.8, 2.7, 4.1, 1.0], [5.6, 3.0, 4.1, 1.3], [5.0, 2.0, 3.5, 1.0], [6.5, 2.8, 4.6, 1.5], [6.1, 2.9, 4.7, 1.4], [6.3, 2.5, 4.9, 1.5], [6.2, 2.2, 4.5, 1.5], [6.0, 3.4, 4.5, 1.6], [5.6, 2.
9, 3.6, 1.3], [5.5, 2.5, 4.0, 1.3], [6.1, 2.8, 4.7, 1.2], [6.3, 2.5, 5.0, 1.9], [6.1, 3.0, 4.9, 1.8], [6.6, 3.0, 4.4, 1.4], [6.0, 2.2, 5.0, 1.5], [5.6, 3.0, 4.5, 1.5], [5.6, 2.5, 3.9, 1.1], [6.7, 3.1, 4.4, 1
.4], [5.5, 2.4, 3.8, 1.1], [6.0, 2.7, 5.1, 1.6], [5.7, 2.8, 4.1, 1.3], [5.8, 2.6, 4.0, 1.2], [5.5, 2.4, 3.7, 1.0], [6.6, 2.9, 4.6, 1.3], [6.3, 2.3, 4.4, 1.3], [6.2, 2.9, 4.3, 1.3], [5.8, 2.7, 5.1, 1.9], [5.7
, 2.9, 4.2, 1.3], [6.4, 3.2, 4.5, 1.5], [6.0, 3.0, 4.8, 1.8], [5.7, 3.0, 4.2, 1.2], [5.9, 3.2, 4.8, 1.8], [5.6, 2.8, 4.9, 2.0], [5.6, 2.7, 4.2, 1.3], [5.4, 3.0, 4.5, 1.5], [6.0, 2.2, 4.0, 1.0], [5.7, 2.6, 3.
5, 1.0], [7.0, 3.2, 4.7, 1.4], [6.0, 2.9, 4.5, 1.5], [6.3, 2.8, 5.1, 1.5], [5.9, 3.0, 4.2, 1.5], [5.2, 2.7, 3.9, 1.4], [6.2, 2.8, 4.8, 1.8], [5.7, 2.5, 5.0, 2.0], [6.1, 2.8, 4.0, 1.3], [5.8, 2.7, 3.9, 1.2],
[5.5, 2.3, 4.0, 1.3], [6.3, 2.7, 4.9, 1.8], [5.7, 2.8, 4.5, 1.3], [5.9, 3.0, 5.1, 1.8], [6.8, 2.8, 4.8, 1.4], [6.4, 2.9, 4.3, 1.3], [6.1, 3.0, 4.6, 1.4], [5.5, 2.6, 4.4, 1.2], [5.8, 2.8, 5.1, 2.4], [5.8, 2.7
, 5.1, 1.9], [6.7, 3.1, 4.7, 1.5], [6.3, 3.3, 4.7, 1.6]]
minimal SSE in kmeans = 78.940841426146

(process:8853): Gtk-WARNING **: 16:49:07.660: Locale not supported by C library.
        Using the fallback 'C' locale.

(python:8853): Gtk-WARNING **: 16:49:07.677: Theme parsing error: gtk.css:3611:22: 'none' is not a valid color name
qt.qpa.xcb: QXcbConnection: XCB error: 5 (BadAtom), sequence: 685, resource id: 0, major code: 20 (GetProperty), minor code: 0
```

```
            finalClusters["cluster1"] = clusterList["cluster1"]
            finalClusters["cluster2"] = clusterList["cluster2"]
            finalClusters["cluster3"] = clusterList["cluster3"]
        else:
            sseValues.append(sseValues[-1])

    print("minimalSSE in bisecting k means= " + str(minimalSSE))
    for cluster in finalClusters:
        print(cluster)
        print(finalClusters[cluster])

    gtk.plotSSEGraphToCompare(sseValues)
    gtk.plot4DGraph(finalClusters)
    gtk.plot3DGraph(finalClusters)
    gtk.plot2DGraph(finalClusters)
    gtk.plotSSEGraph(sseValues)
```

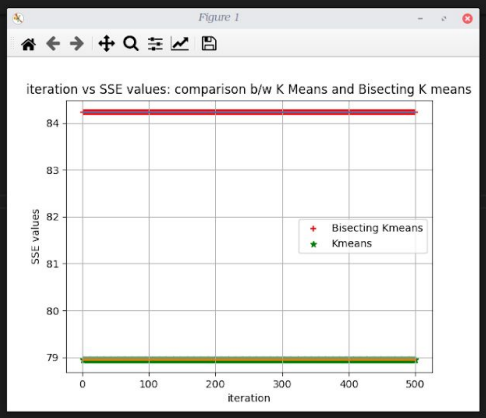


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: python

```
, 3.4, 1.6, 0.2], [5.0, 3.0, 1.6, 0.2], [5.1, 3.8, 1.6, 0.2], [4.9, 3.1, 1.5, 0.1], [5.1, 3.8, 1.9, 0.4], [5.4, 3.9, 1.3, 0.4], [5.0, 2.3, 3.3, 1.0], [5.1, 3.5, 1.4, 0.3], [4.9, 3.1, 1.5, 0.1], [4.4, 2.9, 1.
4, 0.2], [5.1, 3.7, 1.5, 0.4], [5.5, 4.2, 1.4, 0.2], [4.4, 3.2, 1.3, 0.2], [5.1, 2.5, 3.0, 1.1], [4.8, 3.4, 1.9, 0.2], [4.9, 3.1, 1.5, 0.1], [4.4, 3.0, 1.3, 0.2], [4.3, 3.0, 1.1, 0.1], [5.2, 3.5, 1.5, 0.2],
[4.6, 3.6, 1.0, 0.2], [5.0, 3.5, 1.3, 0.3], [5.1, 3.4, 1.5, 0.2], [5.0, 3.3, 1.4, 0.2], [5.3, 3.7, 1.5, 0.2], [4.6, 3.2, 1.4, 0.2]]
cluster3
[[4.9, 2.5, 4.5, 1.7], [5.8, 2.7, 4.1, 1.0], [5.6, 3.0, 4.1, 1.3], [5.0, 2.0, 3.5, 1.0], [6.5, 2.8, 4.6, 1.5], [6.1, 2.9, 4.7, 1.4], [6.3, 2.5, 4.9, 1.5], [6.2, 2.2, 4.5, 1.5], [6.0, 3.4, 4.5, 1.6], [5.6, 2.
9, 3.6, 1.3], [5.5, 2.5, 4.0, 1.3], [6.1, 2.8, 4.7, 1.2], [6.3, 2.5, 5.0, 1.9], [6.1, 3.0, 4.9, 1.8], [6.6, 3.0, 4.4, 1.4], [6.0, 2.2, 5.0, 1.5], [5.6, 3.0, 4.5, 1.5], [5.6, 2.5, 3.9, 1.1], [6.7, 3.1, 4.4, 1
.4], [5.5, 2.4, 3.8, 1.1], [6.0, 2.7, 5.1, 1.6], [5.7, 2.8, 4.1, 1.3], [5.8, 2.6, 4.0, 1.2], [5.5, 2.4, 3.7, 1.0], [6.6, 2.9, 4.6, 1.3], [6.3, 2.3, 4.4, 1.3], [6.2, 2.9, 4.3, 1.3], [5.8, 2.7, 5.1, 1.9], [5.7
, 2.9, 4.2, 1.3], [6.4, 3.2, 4.5, 1.5], [6.0, 3.0, 4.8, 1.8], [5.7, 3.0, 4.2, 1.2], [5.9, 3.2, 4.8, 1.8], [5.6, 2.8, 4.9, 2.0], [5.6, 2.7, 4.2, 1.3], [5.4, 3.0, 4.5, 1.5], [6.0, 2.2, 4.0, 1.0], [5.7, 2.6, 3.
5, 1.0], [7.0, 3.2, 4.7, 1.4], [6.0, 2.9, 4.5, 1.5], [6.3, 2.8, 5.1, 1.5], [5.9, 3.0, 4.2, 1.5], [5.2, 2.7, 3.9, 1.4], [6.2, 2.8, 4.8, 1.8], [5.7, 2.5, 5.0, 2.0], [6.1, 2.8, 4.0, 1.3], [5.8, 2.7, 3.9, 1.2],
[5.5, 2.3, 4.0, 1.3], [6.3, 2.7, 4.9, 1.8], [5.7, 2.8, 4.5, 1.3], [5.9, 3.0, 5.1, 1.8], [6.8, 2.8, 4.8, 1.4], [6.4, 2.9, 4.3, 1.3], [6.1, 3.0, 4.6, 1.4], [5.5, 2.6, 4.4, 1.2], [5.8, 2.8, 5.1, 2.4], [5.8, 2.7
, 5.1, 1.9], [6.7, 3.1, 4.7, 1.5], [6.3, 3.3, 4.7, 1.6]]
minimal SSE in kmeans = 78.940841426146

(process:8853): Gtk-WARNING **: 16:49:07.660: Locale not supported by C library.
        Using the fallback 'C' locale.

(python:8853): Gtk-WARNING **: 16:49:07.677: Theme parsing error: gtk.css:3611:22: 'none' is not a valid color name
qt.qpa.xcb: QXcbConnection: XCB error: 5 (BadAtom), sequence: 685, resource id: 0, major code: 20 (GetProperty), minor code: 0
```

bisecting_kmeans.py > ...

```
    newSSE = findSSE(clusterList, clusterListLen)

    if newSSE < minimalSSE or i == 0:
        minimalSSE = newSSE
        sseValues.append(newSSE)

        finalClusters["cluster1"] = clusterList["cluster1"]
        finalClusters["cluster2"] = clusterList["cluster2"]
        finalClusters["cluster3"] = clusterList["cluster3"]
    else:
        sseValues.append(sseValues[-1])

print("minimalSSE in bisecting k means= " + str(minimalSSE))
for cluster in finalClusters:
    print(cluster)
    print(finalClusters[cluster])

gtk.plotSSEGraphToCompare(sseValues, 500)
# gtk.plot4DGraph(finalClusters)
# gtk.plot3DGraph(finalClusters)
# gtk.plot2DGraph(finalClusters)
gtk.plotSSEGraph(sseValues, 500)
```



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: python

```
, 3.2, 1.3, 0.2], [5.1, 3.5, 1.4, 0.2], [5.4, 3.4, 1.5, 0.4], [4.3, 3.0, 1.1, 0.1], [5.8, 4.0, 1.2, 0.2], [5.4, 3.4, 1.7, 0.2], [4.6, 3.4, 1.4, 0.3], [4.5, 2.3, 1.3, 0.3], [5.1, 3.4, 1.5, 0.2], [5.4, 3.9, 1.7, 0.4], [4.9, 3.1, 1.5, 0.1], [5.1, 3.8, 1.6, 0.2], [4.4, 3.2, 1.3, 0.2], [5.5, 3.5, 1.3, 0.2], [4.4, 2.9, 1.4, 0.2], [5.1, 3.7, 1.5, 0.4], [5.0, 3.0, 1.6, 0.2], [5.5, 4.2, 1.4, 0.2], [5.0, 3.4, 1.6, 0.4], [5.0, 3.2, 1.2, 0.2], [5.0, 2.3, 3.3, 1.0], [5.4, 3.7, 1.5, 0.2], [5.2, 3.4, 1.4, 0.2], [5.0, 3.6, 1.4, 0.2], [5.0, 3.4, 1.5, 0.2]]
cluster3
[[5.8, 2.8, 5.1, 2.4], [5.9, 3.0, 5.1, 1.8], [5.6, 2.5, 3.9, 1.1], [6.0, 3.0, 4.8, 1.8], [7.0, 3.2, 4.7, 1.4], [5.5, 2.4, 3.7, 1.0], [5.5, 2.4, 3.8, 1.1], [6.3, 3.3, 4.7, 1.6], [6.8, 2.8, 4.8, 1.4], [5.6, 2.9, 3.6, 1.3], [6.1, 3.0, 4.6, 1.4], [6.1, 2.9, 4.7, 1.4], [5.5, 2.5, 4.0, 1.3], [5.6, 3.0, 4.5, 1.5], [6.1, 2.8, 4.7, 1.2], [5.5, 2.6, 4.4, 1.2], [6.3, 2.3, 4.4, 1.3], [6.2, 2.2, 4.5, 1.5], [5.6, 2.7, 4.2, 1.3], [5.7, 2.8, 4.1, 1.3], [5.8, 2.7, 5.1, 1.9], [6.3, 2.5, 5.0, 1.9], [5.7, 2.8, 4.5, 1.3], [6.6, 3.0, 4.4, 1.4], [6.0, 2.2, 5.0, 1.5], [4.9, 2.5, 4.5, 1.7], [6.1, 3.0, 4.9, 1.8], [5.8, 2.7, 5.1, 1.9], [6.2, 2.9, 4.3, 1.3], [5.7, 2.6, 3.5, 1.0], [5.8, 2.7, 4.1, 1.0], [6.3, 2.5, 4.9, 1.5], [5.4, 3.0, 4.5, 1.5], [5.7, 2.5, 5.0, 2.0], [6.7, 3.1, 4.7, 1.5], [6.4, 3.2, 4.5, 1.5], [6.7, 3.1, 4.4, 1.4], [6.6, 2.9, 4.6, 1.3], [5.9, 3.0, 4.2, 1.5], [5.7, 3.0, 4.2, 1.2], [5.9, 3.2, 4.8, 1.8], [6.3, 2.8, 5.1, 1.5], [6.0, 2.9, 4.5, 1.5], [5.8, 2.6, 4.0, 1.2], [5.2, 2.7, 3.9, 1.4], [5.5, 2.3, 4.0, 1.3], [5.6, 3.0, 4.1, 1.3], [5.8, 2.7, 3.9, 1.2], [6.0, 2.7, 5.1, 1.6], [6.3, 2.7, 4.9, 1.8], [6.5, 2.8, 4.6, 1.5], [6.2, 2.8, 4.8, 1.8], [6.4, 2.9, 4.3, 1.3], [6.1, 2.8, 4.0, 1.3], [6.0, 2.2, 4.0, 1.0], [5.7, 2.9, 4.2, 1.3], [6.0, 3.4, 4.5, 1.6], [5.0, 2.0, 3.5, 1.0], [5.6, 2.8, 4.9, 2.0]]
minimal SSE in kmeans = 78.94084142614598

(process:4009): Gtk-WARNING **: 09:22:31.726: Locale not supported by C library.
        Using the fallback 'C' locale.
```