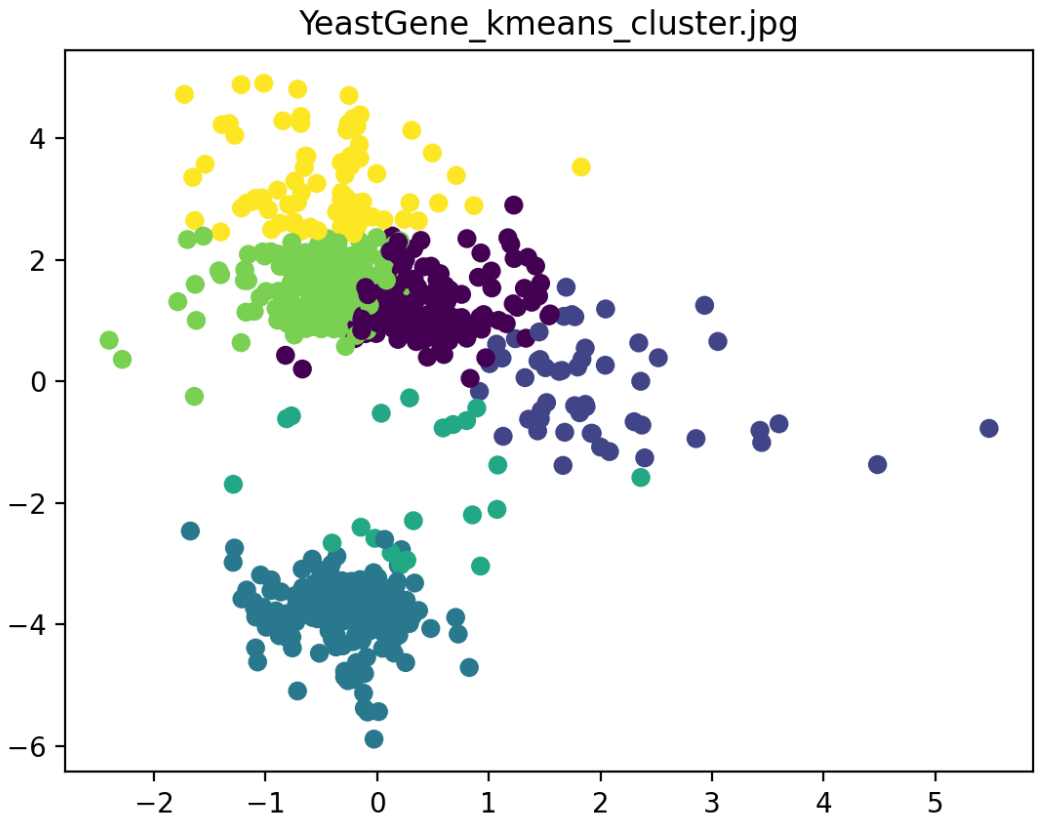
Vivek Khanolkar
ECE 49595 - Assignment 3

1)

```
(base) pal-nat186-47-153:Assignment3 vivek$ python3 kmeans_template.py YeastGene.csv YeastGene_Initial_Centroids.csv 6
T:  7 k:  6
[array([[-0.24127143, -0.12875    ,  0.06225    ,  0.17335    ,  0.21791429,
         1.65169286,  1.90532143]]), array([[-0.9535098 , -1.47164706,  0.07752941, -0.1794902 , -1.00482353,
         1.15121569,  0.96880392]]), array([[ 0.16510366,  0.09165244, -0.10388415, -0.55258537, -0.63008537,
        -1.72318293, -1.75481098]]), array([[ 0.02328571,  0.25080952, -0.27695238, -0.364     , -0.73542857,
        -0.89461905,  0.70014286]]), array([[-1.55555556e-03,  1.56506173e-01,  3.56253086e-01,
         7.01580247e-01,  1.00971605e+00,  1.84231481e+00,
         1.64341975e+00]]), array([[-0.03932895,  0.15394737,  0.43607895,  1.10581579,  1.44871053,
         3.01634211,  2.82938158]])]
(base) pal-nat186-47-153:Assignment3 vivek$
```

2)



3)

```
(base) pal-nat186-47-153:Assignment3 vivek$ python3 Hierarchical_template.py Utilities.csv 1
0-th merging: 12, 21, 23
1-th merging: 10, 13, 24
2-th merging: 4, 24, 25
3-th merging: 7, 23, 26
4-th merging: 25, 20, 27
5-th merging: 14, 19, 28
6-th merging: 1, 18, 29
7-th merging: 15, 26, 30
8-th merging: 29, 28, 31
9-th merging: 2, 27, 32
10-th merging: 8, 16, 33
11-th merging: 32, 30, 34
12-th merging: 34, 22, 35
13-th merging: 9, 31, 36
14-th merging: 35, 36, 37
15-th merging: 6, 37, 38
16-th merging: 3, 38, 39
17-th merging: 39, 33, 40
18-th merging: 17, 40, 41
19-th merging: 11, 41, 42
20-th merging: 5, 42, 43
(base) pal-nat186-47-153:Assignment3 vivek$
```

4)

KMEANS FUNCTIONS:

```
def assignCluster(dataSet, k, centroids):
    '''For each data point, assign it to the closest centroid
    Inputs:
        dataSet: each row represents an observation and
                 each column represents an attribute
        k:  number of clusters
        centroids: initial centroids or centroids of last iteration
    Output:
        clusterAssment: list
            assigned cluster id for each data point
    '''
    #TODO
    clusterAssment = []

    for data in dataSet :
        minDist = float(inf)
        minInd = -1
        for idx, cent in enumerate(centroids) :
            d = distance.euclidean(data, cent)
            if d < minDist :
                minDist = d
                minInd = idx
        clusterAssment.append(minInd)

    return clusterAssment
```

```
def getCentroid(dataSet, k, clusterAssment):
    '''recalculate centroids
    Input:
        dataSet: each row represents an observation and
            each column represents an attribute
        k:  number of clusters
        clusterAssment: list
            assigned cluster id for each data point
    Output:
        centroids: cluster centroids
    '''

    #TODO
    centroids = []
    indData = list(zip(clusterAssment, dataSet))
    for i in range(k):
        centLists = []
        for ind in indData :
            if ind[0] == i :
                centLists.append(ind[1])
        centroids.append(np.array(centLists).mean(axis = 0))
        centroids = list(centroids)

    return centroids
```

HIERARCHICAL FUNCTIONS:

```
def merge_cluster(distance_matrix, cluster_candidate, T):
   ''' Merge two closest clusters according to min distances
   1. Find the smallest entry in the distance matrix—suppose the entry
       is i-th row and j-th column
   2. Merge the clusters that correspond to the i-th row and j-th column
       of the distance matrix as a new cluster with index T

   Parameters:
   ------------
   distance_matrix : 2-D array
       distance matrix
   cluster_candidate : dictionary
       key is the cluster id, value is point ids in the cluster
   T: int
       current cluster index

   Returns:
   ------------
   cluster_candidate: dictionary
       upadted cluster dictionary after merging two clusters
       key is the cluster id, value is point ids in the cluster
```

```
    merge_list : list of tuples
        records the two old clusters' id and points that have just been merged.
        [(cluster_one_id, point_ids_in_cluster_one),
         (cluster_two_id, point_ids_in_cluster_two)]
    '''

    # TODO
    merge_list = []

    minValue = np.amin(distance_matrix)
    index = np.where(distance_matrix == np.amin(distance_matrix))
    i = index[0]
    j = index[1]
    # print("matrix: ", distance_matrix)
    if len(i) > 1:
        a = i[0]
        b = i[1]
    else :
        a = i[0]
        b = j[0]
    # print("a: ", a, "b: ", b)
    # print("min: ", minValue)
    # print("cluster Before: ", cluster_candidate)
    # print("mergeList: ", merge_list)
    for k,v in cluster_candidate.items() :
        for x in v:
            if x == a:
                # print("Key: ", k, "Val: ", v)
                pop1 = k
                val1 = v

    for k,v in cluster_candidate.items() :
        for x in v:
            if x == b:
                # print("Key: ", k, "Val: ", v)
                pop2 = k
                val2 = v

    merge_list.append(tuple(((pop1, val1))))
    merge_list.append(tuple((pop2, val2)))

    if pop1 != pop2:
```

```
        cluster_candidate.pop(pop1)
        cluster_candidate.pop(pop2)
        temp = []
        for oney in val1:
            temp.append(oney)
        for twoy in val2:
            temp.append(twoy)

        cluster_candidate[T] = temp

    # print("cluster After: ", cluster_candidate)
    return cluster_candidate, merge_list
```

```python
def update_distance(distance_matrix, cluster_candidate, merge_list):
    ''' Update the distantce matrix

    Parameters:
    ------------
    distance_matrix : 2-D array
        distance matrix
    cluster_candidate : dictionary
        key is the updated cluster id, value is a list of point ids in the cluster
    merge_list : list of tuples
        records the two old clusters' id and points that have just been merged.
        [(cluster_one_id, point_ids_in_cluster_one),
         (cluster_two_id, point_ids_in_cluster_two)]

    Returns:
    ------------
    distance_matrix: 2-D array
        updated distance matrix
    '''

    # TODO
    # print("distanceMatrix: ", distance_matrix)
    x = merge_list[0][1]
    y = merge_list[1][1]
    # i = x[0]
    # j = y[0]
    for i in x:
        for j in y:
            # print("i: ", i, "j: ", j)
            distance_matrix[i][j] = 100000
            distance_matrix[j][i] = 100000

    return distance_matrix
```