CSE 601
Data Mining and Bioinformatics

Project 2
Clustering Algorithms

Part 1: K-Means Clustering
Part 2: Hierarchical Agglomerative Clustering
Part 3: Density Based Clustering (DBSCAN)
Part 4: Gaussian Mixture Model Clustering
Part 5: Spectral Clustering

Submitted By:
Karan Manchandia
UB Name: karanman
Person # 50290755

Divya Srivastava
UB Name: divyasri
Person # 50290383

Varsha Lakshman
UB Name: varshala
Person # 50288138

Objective:

The objective of this project is to perform the following five clustering algorithms on the two datasets, “cho.txt” and “iyer.txt”:

- K-Means Clustering
- Hierarchical Agglomerative Clustering Using Min approach
- Density Based Clustering i.e. DBSCAN
- Gaussian Mixture Model Clustering (GMM)
- Spectral Clustering

For validation the following two methods are applied:

- Rand Index and Jaccard Coefficient
- Visualize the data using PCA.

Input file details are as follows:

- Each row represents a gene:
- the first column is gene_id.
- the second column is the ground truth clusters. You can compare it with your results. "-1" means outliers.
- the rest columns represent gene's expression values (attributes).

Description of Rand and Jaccard Coefficient (Common to all Algorithms):

- Rand Index and Jaccard Coefficient provide a similarity oriented measure of cluster validity.
- This is based on the premise that the two objects in the same cluster should be in the same class.
- The approach to cluster validity involves comparison of two matrices:
    - Ideal Cluster Similarity Matrix: This matrix has 1 in the $(ij)^{th}$ entry, if two gene objects with gene id i and j are in same cluster, 0 otherwise.
    - Ideal Class Similarity Matrix: this matrix has 1 at the ij^{th} position, if the 2 gene objects with gene id i and j belong to the same class, 0 otherwise.
- So, for calculating rand Index and Jaccard Coefficient we need to create two matrices of the size (no. of gene objects, no. of gene objects) and then populate it according to above defination.
- Then we need to find four different values that are defined as follows:
    - f00: Number of pairs of objects having different class and different clusters.
    - f11: Number of pairs of objects having same class and same clusters.
    - f01: Number of pairs of objects having different class and same clusters.
    - f10: Number of pairs of objects having same class and different clusters.
- Finally, the Rand Index and the Jaccard Coefficient can be calculated as:
    - Rand Index = (| f11 | + | f00 |) / (| f11 | + | f00 | + | f10 | + | f01 |)
    - Jaccard Cofficient= (| f11 |) / (| f11 | + | f10 | + | f01 |)

Algorithm 1: K-Means Clustering

The ***k*-**means clustering algorithm is a data mining and machine learning tool used to cluster observations into groups of related observations without any prior knowledge of those relationships. K-means clustering is a type of unsupervised learning, used with unclassified data. By sampling, the algorithm attempts to show in which category, or cluster, the data belong to, with the number of clusters being defined by the value ***k*.**

Input Parameters:

- File name
- No. of centroids
- Initial centers' indices
- No. of iterations

Algorithm and Implementation Steps:

- Inputting the required parameters, we read the data points from the specified file. The first column corresponds to the gene IDs and the second column are the ground truth values. The remaining columns are the features of that gene id.
- The user is given a choice to input the initial centroids where the initial centroids are fetched using the IDs of the data points provided as input. If the user doesn't wish to provide the initial centroids, then by default the first "K" rows will be used as the K centroid values where "K" is the number of centroids specified by the user.
- The transformed data and inputted centroids along with the number of iterations to be performed are used to compute k mean clusters and final updated centroid values.
- We first divide the original data into clusters by computing the Euclidean distance of each point in original data with the input centroids. The data points with minimum distance from one centroid belong to that particular cluster stated by the centroid. It also maintains the indices of each data point
- Now we take the mean of all the data points per cluster and recalculate the cluster centroids and do the reassignment of the data points as per the new centroids.
- The above process is repeated until convergence, in our case we decide this convergence until the number of iterations inputted by the user. The final cluster indices are used for verifying results using jaccard co-efficient / rand index.
- Using the original data and final cluster indices computed in the above process, are used to compute the jaccard co-efficient and rand index.

Dimensionality Reduction of given dataset:

- For dimensionality reduction of given dataset to 2 dimensions we use sklearn.decomposition PCA library function. We passed the original data into the PCA function and get the new factors.
- Plot the new factors on a matplotlib plot and color the point according to the cluster the gene id belongs to.
- Take the second column from the original data file and use it as ground_truth.
- Define two numpy matrix (Ideal Cluster Similarity Matrix and Ideal class Similarity Matrix) of the size (no. of gene objects, no. of gene objects). Populate the Ideal Cluster Similarity Matrix such that it has 1 in the (ij)th entry, if two gene objects with gene id i and j are in same cluster,

0 otherwise. Similarly, populate the ideal class similarity matrix with 1 at the (ij)th position, if the 2 gene objects with gene id i and j belong to the same class, 0 otherwise.

- Now, calculate four different values that are defined as follows:
  - f00: Number of pairs of objects having different class and different clusters.
  - f11: Number of pairs of objects having same class and same clusters.
  - f01: Number of pairs of objects having different class and same clusters.
  - f10: Number of pairs of objects having same class and different clusters.
- Calculate and print the Jaccard Cofficient and Rand Index where:
  - Jaccard = f11/(f11+f10+f01)
  - Rand = (f11+f00)/(f11+f10+f01+f00).

Results:

Input-1(Cho.txt):

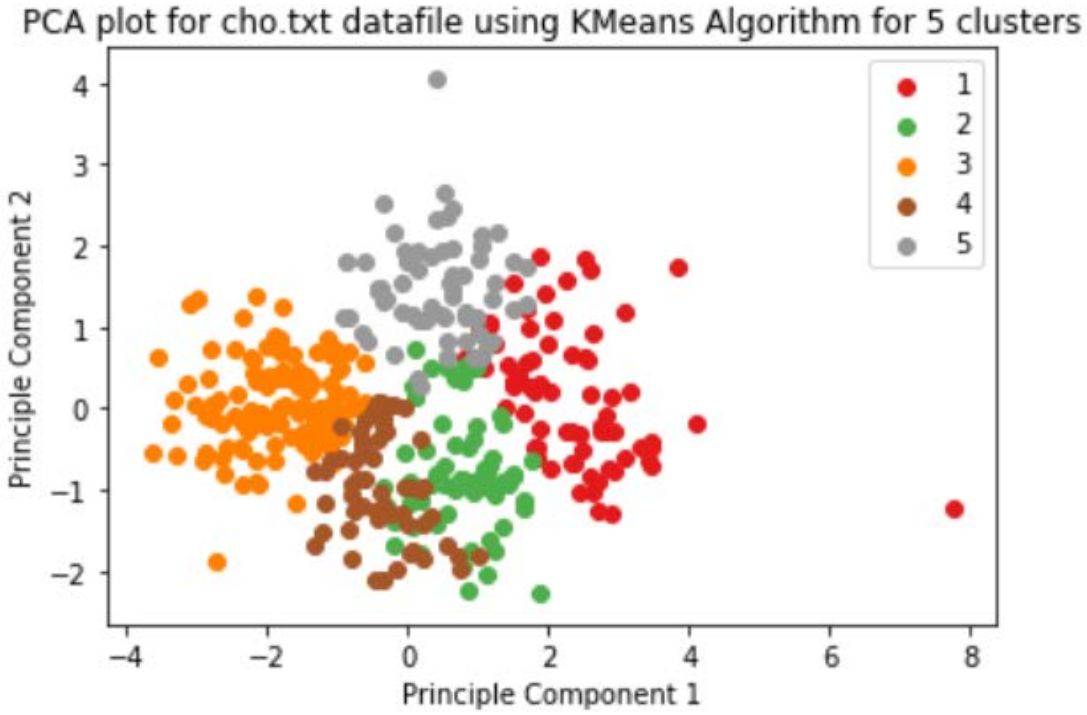
- Enter the name of the gene expression data file: cho.txt
- Do you want to enter initial centroids (Y or N): Y
- Enter initial centers indices: 5,25,32,100,132
- No. of centroids, k = 5
- Enter the total number of iterations: 300

Output-1(Cho.txt):

- Here, the colors correspond to the cluster id assigned by the algorithm and corresponding clusters and data points assignments.
- The RAND coefficient is: 0.8111627157776048
- The JACCARD coefficient is: 0.4064635895705004

Result Screenshot:

```
The file name is: cho.txt
The number of centroids i.e. cluster centers are: 5
No. of iterations are: 300
Initial centroids are:
[[-1.04   0.13   0.51  -0.44  -0.88  -0.32   0.21   0.95   1.07   0.38
   0.01  -0.13  -0.78  -0.13   0.092  0.   ]
 [-0.26   0.24   0.61   0.34   0.57  -0.09  -0.13  -0.34  -0.46   0.19
   0.026 -0.42  -0.17  -0.29  -0.05  -0.14 ]
 [-0.45  -0.1    1.04   0.44   0.07   0.27   0.17  -0.33  -0.37   0.41
   0.17   0.04  -0.078 -0.24  -0.67  -0.76 ]
 [-0.64  -0.42   0.58   0.66   0.52  -0.02  -0.42  -0.68  -0.44   0.29
   0.51   0.163  0.23  -0.12  -0.22  -0.32 ]
 [ 0.05   0.05   0.86   0.46  -0.6   -0.62  -0.92  -0.86  -0.35   0.65
   0.19   0.16   0.12  -0.089  0.22   0.27 ]]
New centroids at the end of 300 iterations are:
[[-0.69263768 -0.72352174 -0.93565217 -0.6884058  -0.32913043  0.08753623
   0.62565217  0.94        0.83217391 -0.02362319 -0.33086957 -0.19652174
  -0.00452174  0.31137681  0.51223188  0.47246377]
 [-0.31408451 -0.19543662 -0.26487324  0.07585915  0.39707042  0.4103662
   0.30292958  0.16802817 -0.04749296 -0.45692958 -0.14998592  0.12392958
   0.22285915  0.10214085  0.17026761 -0.06971831]
 [-0.28854098  0.13377049  1.1752459   0.59959016 -0.05885246 -0.235
  -0.55467213 -0.79845902 -0.60795082  0.73196721  0.60886885  0.24598361
  -0.06507377 -0.37268033 -0.59832787 -0.41581967]
 [-0.77610169 -0.50966102  0.36864407  0.5579661   0.55016949  0.25627119
  -0.00152542 -0.27745763 -0.38627119  0.16152542  0.33113559  0.2829661
   0.1410339   0.01057627 -0.14771186 -0.26949153]
 [ 0.09106154 -0.25230769 -0.60630769 -0.64584615 -0.56676923 -0.41953846
  -0.49461538 -0.37876923  0.18107692  0.84076923  0.53492308  0.24838462
   0.02470769  0.05869231  0.09210769  0.44292308]]
The RAND coefficient is: 0.8111627157776048
The JACCARD coefficient is: 0.4064635895705004
```



Input-2(Cho.txt):

- Enter the name of the gene expression data file: cho.txt
- Do you want to enter initial centroids (Y or N): N
- Enter the number of centroids i.e. k: 5
- No. of centroids, k = 5
- Enter the total number of iterations: 300

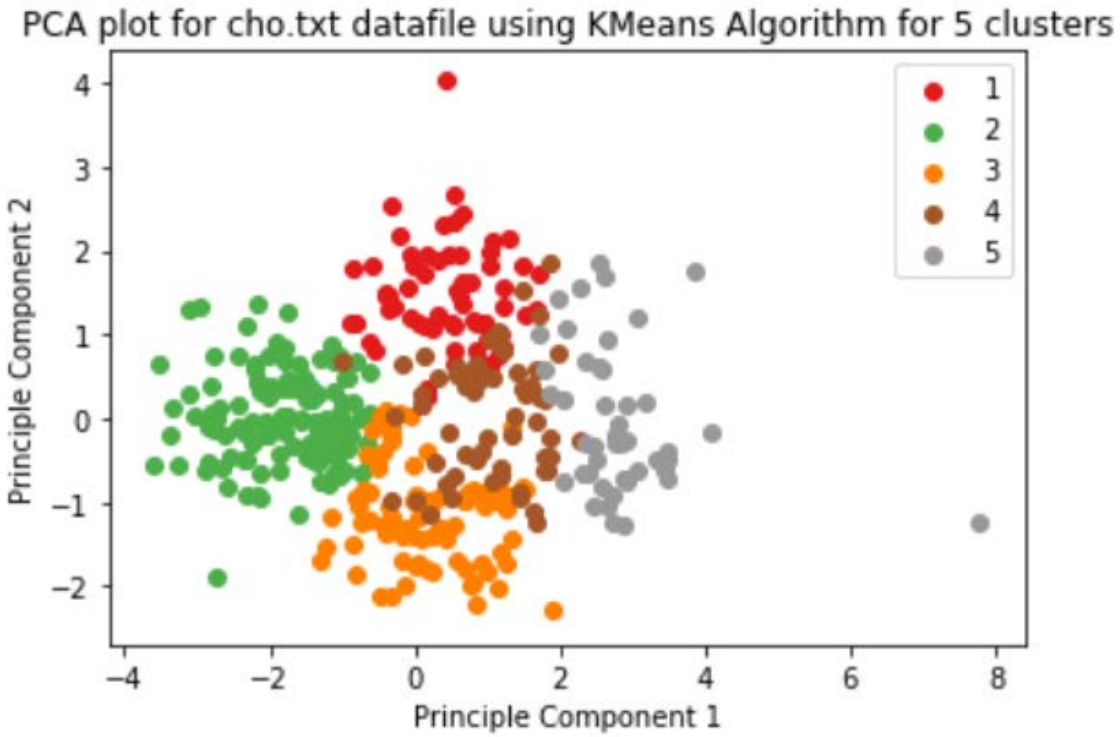
Since the initial centroids are not specified, the function will select the first k rows from the dataset and set them as initial centroid values.

Output-2(Cho.txt):

- Here, the colors correspond to the cluster id assigned by the algorithm and corresponding clusters and data points assignments.
- The RAND coefficient is: 0.7917527987328519
- The JACCARD coefficient is: 0.37609587388401833

Result Screenshot:

```
The file name is: cho.txt
The number of centroids i.e. cluster centers are: 5
No. of iterations are: 300
Initial centroids are:
[[-0.69  -0.96  -1.16  -0.66  -0.55   0.12  -1.07  -1.22   0.82   1.4
   0.71   0.68   0.11  -0.04   0.19   0.82 ]
 [-0.21   0.19   0.86   0.04  -0.35  -0.39  -0.51  -0.2    0.     0.77
   0.41   0.14  -0.45  -1.23  -0.325  0.   ]
 [-0.3   -0.56  -0.29  -0.5   -0.27  -0.29  -0.56  -1.04   0.32   0.9
   0.45   0.17   0.164 -0.12  -0.16   0.67 ]
 [ 0.07   0.26  -0.47  -0.68  -0.63  -0.39   0.07   0.79   0.58   0.31
  -0.14  -0.29  -0.103 -0.2   -0.06   0.36 ]
 [-1.04   0.13   0.51  -0.44  -0.88  -0.32   0.21   0.95   1.07   0.38
   0.01  -0.13  -0.78  -0.13   0.092  0.   ]]
New centroids at the end of 300 iterations are:
[[ 0.10363492 -0.27095238 -0.62761905 -0.65174603 -0.57730159 -0.42666667
  -0.50333333 -0.38238095  0.17603175  0.86333333  0.54619048  0.26230159
   0.02596825  0.06047619  0.10693651  0.44142857]
 [-0.3383876   0.09426357  1.14821705  0.59713178 -0.03643411 -0.21465116
  -0.53054264 -0.78094574 -0.59837209  0.72620155  0.62110078  0.25756589
  -0.05465891 -0.35779845 -0.58712403 -0.41775194]
 [-0.69730337 -0.51685393  0.02988764  0.41337079  0.60146067  0.44044944
   0.16505618 -0.09764045 -0.28842697 -0.13955056  0.12839326  0.27932584
   0.2601236   0.09388764  0.01180899 -0.20662921]
 [-0.02633333  0.0654     -0.26593333 -0.2609     -0.07796667  0.01793333
   0.24913333  0.43333333  0.32746667 -0.24436667 -0.26081667 -0.2252
  -0.11121667  0.02171667  0.1748      0.16483333]
 [-0.98671111 -1.04362222 -1.17622222 -0.78066667 -0.308       0.19222222
   0.784       1.02022222  0.91066667 -0.03511111 -0.33777778 -0.06688889
   0.14302222  0.45324444  0.64826667  0.542     ]]
The RAND coefficient is: 0.7917527987328519
The JACCARD coefficient is: 0.37609587388401833
```



Input-3(iyer.txt):

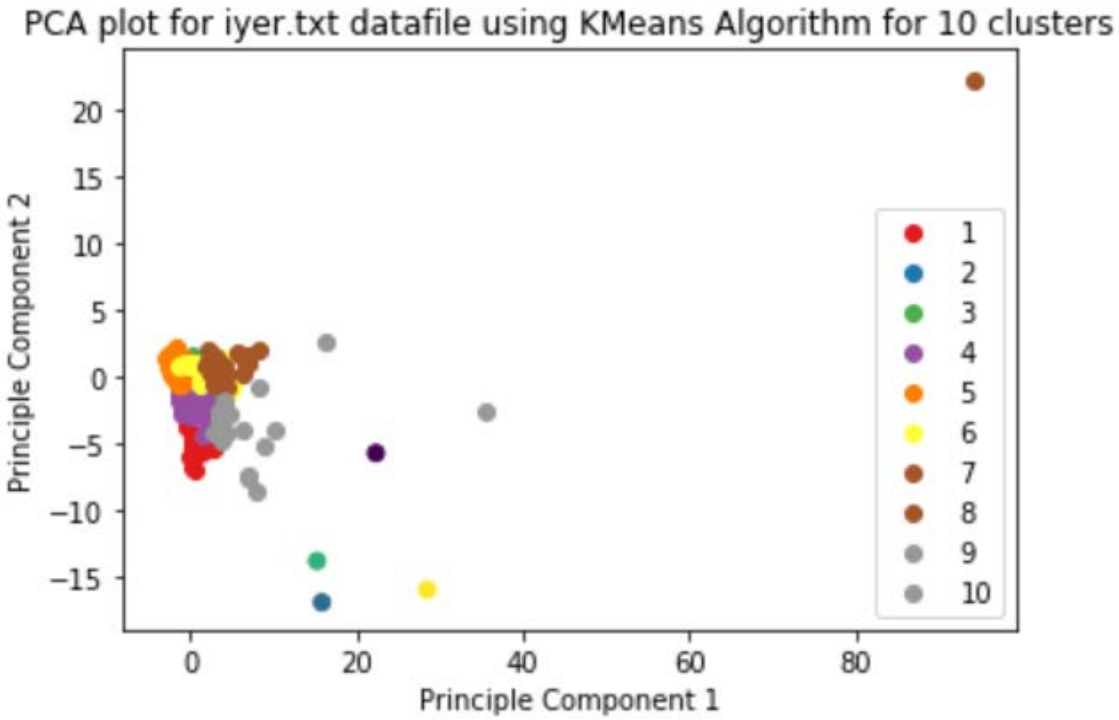
- Enter the name of the gene expression data file: iyer.txt
- Do you want to enter initial centroids (Y or N): Y
- Enter initial centers indices: 45,2,56,23,1,67,3,11,5,89
- No. of centroids, k = 10
- Total number of iterations is set to be 300

Output-3:

- Here, the colors correspond to the cluster id assigned by the algorithm and corresponding clusters and data points assignments.
- The RAND coefficient is: 0.7201830228703763
- The JACCARD coefficient is: 0.3187908154436075

Output Screenshots:

```
The file name is: iyer.txt
The number of centroids i.e. cluster centers are: 10
No. of iterations are: 300
Initial centroids are:
[[1.   0.82 0.85 0.82 0.75 0.55 0.57 0.37 0.5  0.71 0.53 0.65]
 [1.   1.58 1.05 1.15 1.22 0.54 0.73 0.82 0.82 0.9  0.73 0.75]
 [1.   0.73 1.31 0.85 0.89 0.6  0.57 0.41 0.41 0.57 0.43 0.4 ]
 [1.   0.91 0.86 0.73 1.14 0.67 0.65 0.44 0.42 0.75 0.48 0.54]
 [1.   0.72 0.1  0.57 1.08 0.66 0.39 0.49 0.28 0.5  0.66 0.52]
 [1.   0.93 1.2  0.81 0.97 0.78 0.69 0.63 0.41 0.41 0.32 0.33]
 [1.   1.1  0.97 1.   0.9  0.67 0.81 0.88 0.77 0.71 0.57 0.46]
 [1.   1.12 0.92 1.01 0.86 0.86 0.7  0.62 0.36 0.37 0.35 0.38]
 [1.   1.21 1.29 1.08 0.89 0.88 0.66 0.85 0.67 0.58 0.82 0.6 ]
 [1.   1.3  1.27 1.07 1.05 0.64 0.42 0.48 0.61 0.7  0.58 0.57]]
New centroids at the end of 300 iterations are:
[[ 1.          1.17411765  1.30294118  1.15294118  1.21647059  1.07588235
   1.15176471  1.23294118  1.53823529  2.97        4.51352941  6.69647059]
 [ 1.          0.9775      1.1575      1.4225      1.255       5.6975
  12.915      14.8275     10.21       11.495       9.7475      7.5475    ]
 [ 1.          0.90851852  1.21555556  1.38481481  1.2712963   1.86814815
   1.91425926  2.23574074  2.11962963  1.40907407  1.31703704  1.18203704]
 [ 1.          1.01362069  0.98706897  1.06396552  1.11137931  1.20224138
   1.30586207  1.51396552  1.72189655  2.11603448  2.47517241  2.87896552]
 [ 1.          0.95464286  0.9363961   0.9212987   0.89181818  0.70042208
   0.57415584  0.50987013  0.5112013   0.74681818  0.74373377  0.79613636]
 [ 1.          1.77904762  3.4847619   5.69190476  4.04095238  2.68857143
   2.35571429  2.26190476  1.55761905  1.37190476  1.33714286  1.25333333]
 [ 1.          2.67        2.41        3.75        6.18       38.8
  86.92       25.58       14.81        6.73        2.72        2.42      ]
 [ 1.          0.9715625   1.165       1.5190625   1.76375     4.2071875
   4.1021875   3.3628125   1.773125    1.38125     1.3253125   1.2675    ]
 [ 1.          1.415       4.275       8.49        7.67       20.4
  15.255      13.535       6.5         2.225       3.37        4.595     ]
 [ 1.          1.008       1.1305      1.2245      1.4475      2.7075
   3.8785      5.484       4.8625      3.589       3.3575      2.9495    ]]
The RAND coefficient is: 0.7201830228703763
The JACCARD coefficient is: 0.3187908154436075
```



Input-4(iyer.txt):

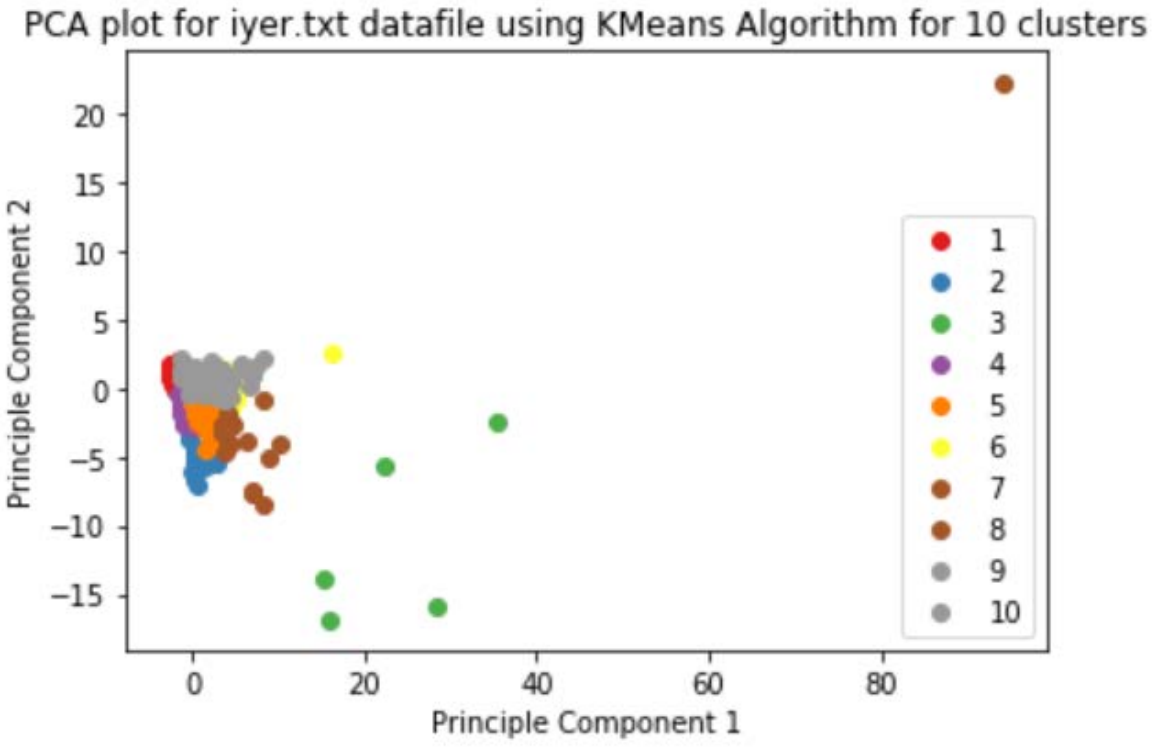
- Enter the name of the gene expression data file: iyer.txt
- Do you want to enter initial centroids (Y or N): N
- Enter the number of centroids i.e. k: 10
- No. of centroids, k = 10
- Enter the total number of iterations: 300

Output-4(iyer.txt):

- Here, the colors correspond to the cluster id assigned by the algorithm and corresponding clusters and data points assignments.
- The RAND coefficient is: 0.7373554467262027
- The JACCARD coefficient is: 0.33329534554640683

Result Screenshot:

```
The file name is: iyer.txt
The number of centroids i.e. cluster centers are: 10
No. of iterations are: 300
Initial centroids are:
[[1.   0.72 0.1  0.57 1.08 0.66 0.39 0.49 0.28 0.5  0.66 0.52]
 [1.   1.58 1.05 1.15 1.22 0.54 0.73 0.82 0.82 0.9  0.73 0.75]
 [1.   1.1  0.97 1.   0.9  0.67 0.81 0.88 0.77 0.71 0.57 0.46]
 [1.   0.97 1.   0.85 0.84 0.72 0.66 0.68 0.47 0.61 0.59 0.65]
 [1.   1.21 1.29 1.08 0.89 0.88 0.66 0.85 0.67 0.58 0.82 0.6 ]
 [1.   1.45 1.44 1.12 1.1  1.15 0.79 0.77 0.78 0.71 0.67 0.36]
 [1.   1.15 1.1  1.   1.08 0.79 0.98 1.03 0.59 0.57 0.46 0.39]
 [1.   1.32 1.35 1.13 1.   0.91 1.22 1.05 0.58 0.57 0.53 0.43]
 [1.   1.01 1.38 1.21 0.79 0.85 0.78 0.73 0.64 0.58 0.43 0.47]
 [1.   0.85 1.03 1.   0.81 0.82 0.73 0.51 0.24 0.54 0.43 0.51]]
New centroids at the end of 300 iterations are:
[[ 1.          0.94831126  0.9315894   0.91410596  0.87890728  0.69483444
   0.57350993  0.51152318  0.51029801  0.74112583  0.73682119  0.78221854]
 [ 1.          1.17411765  1.30294118  1.15294118  1.21647059  1.07588235
   1.15176471  1.23294118  1.53823529  2.97        4.51352941  6.69647059]
 [ 1.          1.018       2.032       3.518       1.776      10.202
  14.346      15.17       10.084       9.856       8.974       7.704     ]
 [ 1.          1.09307692  0.91128205  0.96102564  1.00307692  0.94102564
   0.93384615  0.98846154  1.19358974  1.84307692  2.32410256  2.92128205]
 [ 1.          0.89473684  1.08701754  1.13631579  1.13210526  1.71052632
   1.89035088  2.33824561  2.52824561  2.06035088  2.0122807   1.94701754]
 [ 1.          1.8         3.83230769  6.75846154  5.11846154  4.17384615
   3.51230769  3.36615385  1.80846154  1.55769231  1.46        1.33923077]
 [ 1.          1.02105263  1.15        1.24210526  1.46578947  2.75684211
   3.95842105  5.59210526  4.96052632  3.54157895  3.34        2.93578947]
 [ 1.          2.67        2.41        3.75        6.18       38.8
  86.92       25.58       14.81        6.73        2.72        2.42      ]
 [ 1.          0.95972973  1.17918919  1.49297297  1.72108108  3.97162162
   3.92405405  3.33135135  1.78864865  1.35027027  1.2927027   1.22351351]
 [ 1.          1.25222222  1.97222222  2.70185185  2.31        1.72333333
   1.52851852  1.53037037  1.22925926  0.89407407  0.89333333  0.85740741]]
The RAND coefficient is: 0.7373554467262027
The JACCARD coefficient is: 0.33329534554640683
```



Result Evaluation:

- There is a variation in the observed Rand and Jaccard values when we select the initial centroids in random and when we provide the initial centroid values.
- This signify that sometimes the initial centroids will readjust themselves in ‘right’ way, and sometimes they don’t.

Advantages:

- K-means provides computationally faster results for smaller values of k.
- The results provided by K-means are compact.
- K-means clustering is usually observed to be more efficient as per run-time: O(tkn), where n is # objects, k is # clusters, and t is # iterations.
- For large dataset clustering, k-means provide best results.

Disadvantages:

- It is difficult to predict the value of K without initial ground truth values.
- Different similarity results are obtained on selecting different initial centroids.
- Most of the data points are usually clustered but can also lead to empty clusters.
- For obtaining perfect results, some pre and post processing may be required on the initial and final data.
- For the clusters of different in sizes, densities and shapes, it doesn’t perform well.

Applications:

- Feature extraction from images,videos,text, etc.
- K Means clustering method can be used for anomaly detection in Healthcare Fraud Detection. Can be used in cluster based techniques for plagiarism detection

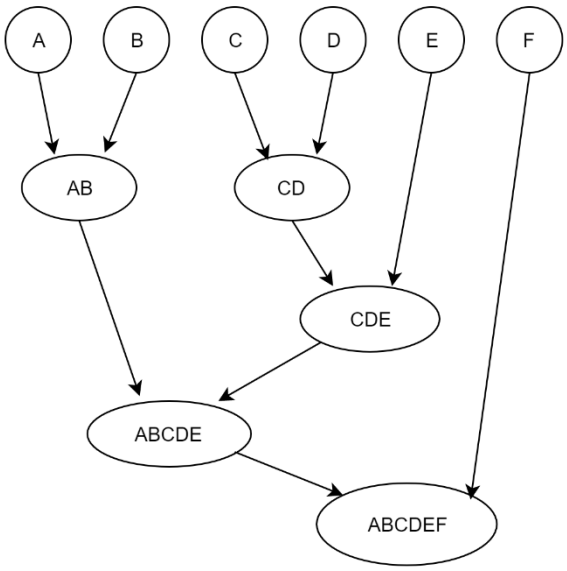
Algorithm 2: Hierarchical Agglomerative Clustering (with Min Approach)

Introducation:

- Hierarchical Agglomerative clustering is a bottom-up strategy to partition a set of data objects into subsets. These subsets are called clusters.
- This clustering method start by letting each object form its own cluster and iteratively merges clusters into larger clusters, until all the data objects are in a single cluster or certain termination condition is met.
- At the beginning of this clustering method each data object belong to a single cluster. Each cluster is recursively merged with its nearest neighbor based on the minimum Euclidean distance, until all the points belong to a single cluster.
- In Min Approach we define cluster proximity as the proximity between two closest points that are in different clusters.

Explanation of Algorithm:

- **Step I :** In the first step we calculate the proximity of each individual point and consider that all the data points belong to the individual clusters. In Min approach the similarity of two clusters C1 and C2 is equal to the minimum of the distance between points p_i and p_j such that p_i belongs to C1 and p_j belongs to C2. Let's say we are given six data points A,B,C,D,E,F. So after this step we would keep each individual point in one cluster.
- **Step II:** In the second step we merge similar cluster and form a single cluster. Say, based on similarity A,B and C,D are similar clusters that are merged in this step. After this step, now we have 4 clusters AB, CD, E and F.
- **Step III:** Now, we calculate proximity of the new clusters and merge similar clusters to form new clusters. Say, in this step CD and E is merged based on similarity to form new clusters AB, CDE and F.
- **Step IV:** Once again calculate the proximity of new clusters and merge similar clusters to form new clusters. Say, now we merged new clusters AB and CDE based on similarity and we are finally left with 2 clusters ABCDE and F.
- **Step V:** Finally, all the clusters are merged together and form a single cluster.



Implementation:

- **Step I:** Take the datafile name as input from the user. Open the tab delimited text file and remove the first 2 clumns (gene id and classes). Store the numpy matrix as a variable ‘impo_data’.

```
Enter the name of the gene expression data file:cho.txt
Correct data file found and imported

------------------------------------------------------------------------

The given data file in a numpy matrix is shown below:

[[-0.69  -0.96  -1.16  -0.66  -0.55   0.12  -1.07  -1.22   0.82   1.4
   0.71   0.68   0.11  -0.04   0.19   0.82 ]
 [-0.21   0.19   0.86   0.04  -0.35  -0.39  -0.51  -0.2    0.     0.77
   0.41   0.14  -0.45  -1.23  -0.325  0.   ]
 [-0.3   -0.56  -0.29  -0.5   -0.27  -0.29  -0.56  -1.04   0.32   0.9
   0.45   0.17   0.164 -0.12  -0.16   0.67 ]
 [ 0.07   0.26  -0.47  -0.68  -0.63  -0.39   0.07   0.79   0.58   0.31
  -0.14  -0.29  -0.103 -0.2   -0.06   0.36 ]
 [-1.04   0.13   0.51  -0.44  -0.88  -0.32   0.21   0.95   1.07   0.38
   0.01  -0.13  -0.78  -0.13   0.092  0.   ]
 [-1.17   0.09  -0.52  -1.04  -1.16  -0.83   0.17   0.93   0.89   0.52
  -0.24  -0.46  -0.215  0.2    0.91   0.68 ]
```

- **Step 2:** As the columns with zero standard deviation and variance are not significant for clustering, remove the columns with zero standard deviation and variance. Note that this step may not be required for the given datafiles for the project, but I have implemented this as this is standard across industry.
- **Step 3:** Take input from the user for the total number of clusters.
- **Step 4:** Calculate the Euclidean distance between each pair of data objects.
    - For this purpose initialize an empty numpy matrix of the size (p * (p-1))/2 where p is the number of data objects in the data file.
    - The number of columns in the numpy matrix should be three, representing the first point, second point and the distance.
    - We use scipy.linalg.norm to calculate the distance between points and populate the distance matrix.

```
The size of the distance matrix is: (74305, 3)


The distance matrix is shown below:

[[  0.          1.          3.44900348]
 [  0.          2.          1.61728662]
 [  0.          3.          3.37681344]
 [  0.          4.          3.81167994]
 [  0.          5.          3.60613713]
 [  0.          6.          3.67523931]
 [  0.          7.          2.96150907]
 [  0.          8.          2.83638855]
 [  0.          9.          3.18155559]
 [  0.         10.          3.20217364]
 [  0.         11.          2.64138903]
 [  0.         12.          1.397355  ]
 [  0.         13.          3.05727411]
 [  0.         14.          2.55844406]
```

- **Step 5:** Sort the distance matrix by distance in ascending order and convert it into a pandas dataframe.
- **Step 6:** Now the first row in the sorted distance matrix corresponds to the closest points. We need to merge these points into a cluster. Then, we start merging clusters till only a certain number of clusters remain. For merging, the points with minimum distance between them is identified from the distance matrix. These points are then merged by updating the list of points and the entry for single point is deleted.
- **Step 7:** Recompute the distance matrix for the remaining points, using the minimum distance between two of the merged points. Recompute the distance matrix in each iteration when two points or clusters are merged. The final output is the lists of gene ids for each clusters. Print the cluster assigned gene ids for each cluster id.
- **Step 8:** Dimensionality Reduction of given dataset. For dimensionality reduction of given dataset to 2 dimensions we use sklearn.decomposition PCA library function. We passed the normalized data into the PCA function and get the new factors.
- **Step 9:** Plot the new factors on a matplotlib plot and color the point according to the cluster the gene id belongs to.
- **Step 10:** Take the second column from the original data file and store it as variable ground_truth.
- **Step 11:**
    - Define two numpy matrix (Ideal Cluster Similarity Matrix and Ideal class Similarity Matrix) of the size (no. of gene objects, no. of gene objects).
    - Populate the Ideal Cluster Similarity Matrix such that it has 1 in the $(ij)^{th}$ entry, if two gene objects with gene id I and j are in same cluster, 0 otherwise.
    - Similarly, populate the ideal class similarity matrix with 1 at the ij^{th} position, if the 2 gene objects with gene id i and j belong to the same class, 0 otherwise.
- **Step 12:** Now, calculate four different values that are defined as follows:
    - f00: Number of pairs of objects having different class and different clusters.
    - f11: Number of pairs of objects having same class and same clusters.
    - f01: Number of pairs of objects having different class and same clusters.
    - f10: Number of pairs of objects having same class and different clusters.
- **Step 13:** Calculate and print the Rand and Jaccard where:
    - Jaccard Cofficient = f11/(f11+f10+f01)
    - Rand Index = (f11+f00)/(f11+f10+f01+f00)

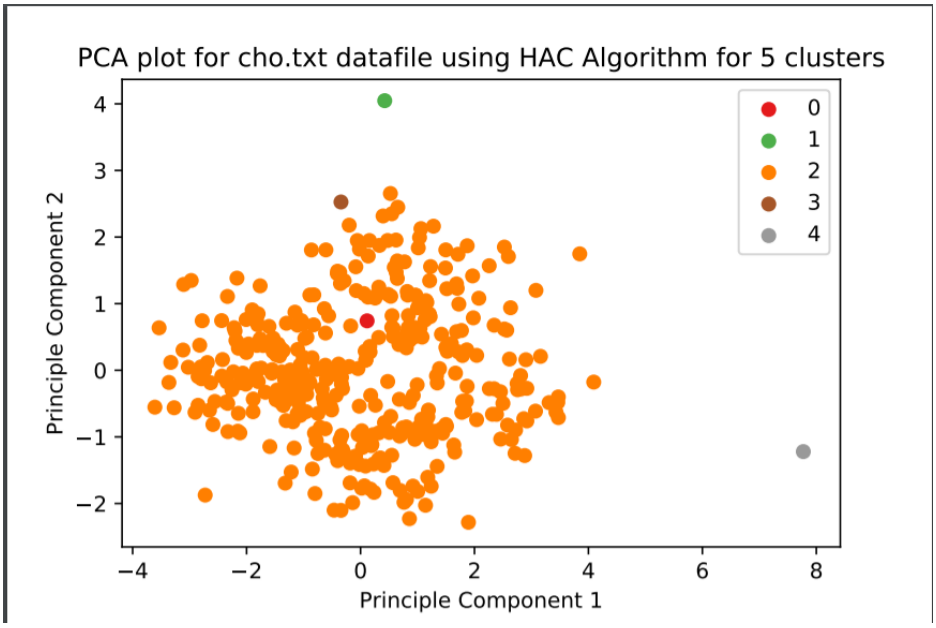
Libraries Used:

- **Sys:** Used to set the numpy matrix to display max size and floating point numbers.
- **Pandas:** Used for pandas dataframe and related operations.
- **Numpy:** Used for numpy matrix and related operations.
- **matplotlib.pyplot:** Used for plotting matplotlib plot
- **sklearn.decomposition(PCA):** Used for performing dimensionality reduction on the dataset.
- **Scipy.linalg(norm):** Used to calculate Eucledean distance.
- **Matplotlib.cm:** Used for defining variable colors for matplotlib plot.

Results:

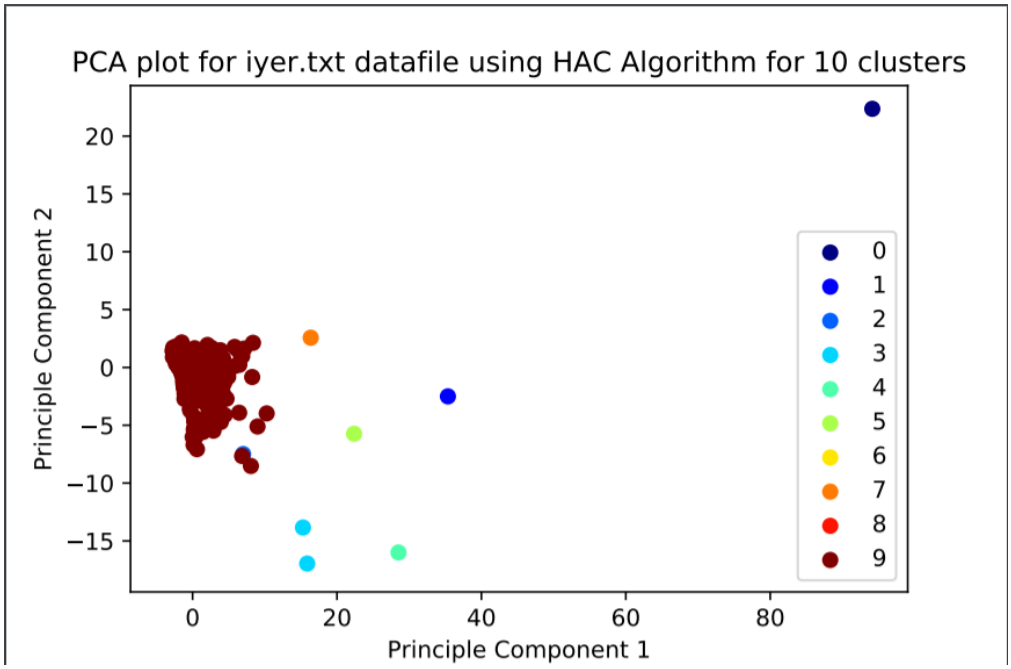
Input File: "cho.txt"

The Jaccard Coefficient for HAC Algorithm(cho.txt) for 5 clusters is: **0.22839497757358454**
The Rand Index for HAC Algorithm(cho.txt) for 5 clusters is: **0.24027490670890495**



Input File: "iyer.txt"

The Jaccard Coefficient for HAC Algorithm(iyer.txt) for 10 clusters is: **0.15414608786067066**
The Rand Index for HAC Algorithm(iyer.txt) for 10 clusters is: **0.18433605572994025**



Result Evaluation:

- We have successfully implemented the Hierarchical Agglomerative Clustering Algorithm using the Min Approach. This is also called as single linkage HAC Clustering.
- By observing the results for the dataset iyer.txt and cho.txt we can say that HAC(min Approach) is capable of handling non elliptical shaped clusters.
- Someone might argue that why a single point that is plotted in between a large cluster belong to a different cluster for “cho” datafile. This may be due to dimensionality reduction. With many dimensions the point may be at a large distance from the large cluster. The same large distance might not be represented after dimensionality reduction.
- From the final plots we can conclude that HAC algorithm does not give evenly distributed clusters, when we input the final number of clusters.

Advantages:

- No information about number of clusters is required as input, though we are taking it as input but it can work without that.
- HAC outputs a hierarchiy that can be represented by using a dendrogram. This is more informative then set of clusters, each cluster containing objects, that we get in k-means.
- It is easier to decide on the number of clusters by seeing the dendrogram. We can quickly get the results for a particular number of clusters by cutting a dendrogram accordingly.
- HAC results may correspond to meaningful Taxonomies.

Disadvantages:

- HAC is sensitive to noise and outliers.
- The order of data can have an impact on final clustering results.
- Once a decision is made to combine two clusters it cannot be undone.
- We have got the results where some clusters have only 1 gene object. So we can say that for datasets like cho and iyer clusters are not evenly distributed.
- The time complexity of the HAC algorithm is $O(n^2)$. So, HAC algorithm is not a good fit for very large datasets.
- If we have a large dataset, it may become difficult to determine the correct number of clusters by using the dendrogram.

Applications:

- Clustering of twitter data related to a particular set of topics.
- Charting evolution through phylogenetic trees
- Grouping of images or documents as per similarity.

Algorithm 3: Density-based spatial clustering of applications with noise (DBSCAN)

Clusters are dense regions in the data space, separated by regions of lower object density. A cluster is defined as a maximal set of density-connected points. DBSCAN is a density-based clustering algorithm. It groups together points that are closely packed together. The points which lie alone in low-density regions are marked as outliers.

Algorithm:

- An arbitrary starting point has to be selected that has not been visited.
- The distance between this point and other points is calculated. If it is equal or less than the epsilon(radius), they can be considered as neighbors.
- If the starting point's number of neighbors is more than the minimum number of points required to form a dense region, a cluster is started or else the point is labelled as noise.
- The points in the cluster's neighbors are retrieved and checked. The neighbors of the points are found. If the point has more than the minimum number of points required in its neighborhood, its ε-neighborhood is also considered to be part of that cluster. Therefore, all points that are found within the ε-neighborhood are also added to the cluster. This process is repeated until a density-connected cluster is formed.
- The point which was earlier pointed as noise can be made part of a cluster, if it is found in the ε-environment of a different point.
- The algorithm runs until all the points are either put into a cluster or are marked as noise.

DBSCAN requires two parameters:

- MinPts: The minimum number of points required to form a dense region.
- Epsilon: It specifies the radius of a neighborhood with respect to a point.

For DBSCAN clustering, the points which are to be clustered are grouped into the following categories:

- Core Points: A point can be considered as a core point, if within the epsilon distance of that point, there are at least minimum number of points mentioned in the parameter. These are points that are at the interior of a cluster.
- Border Points: A point is considered to be a border point, if it has less than the minimum number of points in its neighborhood but it is in the neighborhood of core points.
- Noise or outliers: Points that are not reachable from any other point.

Implementation:

- First, we have given the options to enter the filename, epsilon and the minimum number of points. Then we are reading the file using genfromtxt function of numpy. From this data, we have removed the first two columns.
- In the dbscan function, we have selected a point, if it is not visited, we are calculating the distance of other points with this point. If the other points are within the epsilon distance of our first selected point, the other points are considered to be neighbors of this point.
- If the number of neighbors is greater than the minimum number of points, we are initiating a cluster and labelling that point to its corresponding cluster.
- If the number of neighbors is less than the minimum number of points it is classified as noise.
- We are iterating through the points in the neighborhood, and checking if their neighbors have minimum number of points required to form a dense region. If the neighboring points are

density reachable, then those neighboring points are added to the same cluster. [A point q is said to be density reachable from p if point *q* is within the epsilon distance from core point p].

- Finally, we have labelled all the points according to their corresponding cluster id. If a point doesn't belong to any cluster, we have labelled it as -1. For plotting the results, we have used PCA.
- Dimensionality Reduction of given dataset. For dimensionality reduction of given dataset to 2 dimensions we use sklearn.decomposition PCA library function. We passed the normalized data into the PCA function and get the new factors.
- Plot the new factors on a matplotlib plot and color the point according to the cluster the gene id belongs to.
- Then find the Rand and Jaccard Coefficient. The steps for it is same as in the other algorithms explained above.

Advantages:

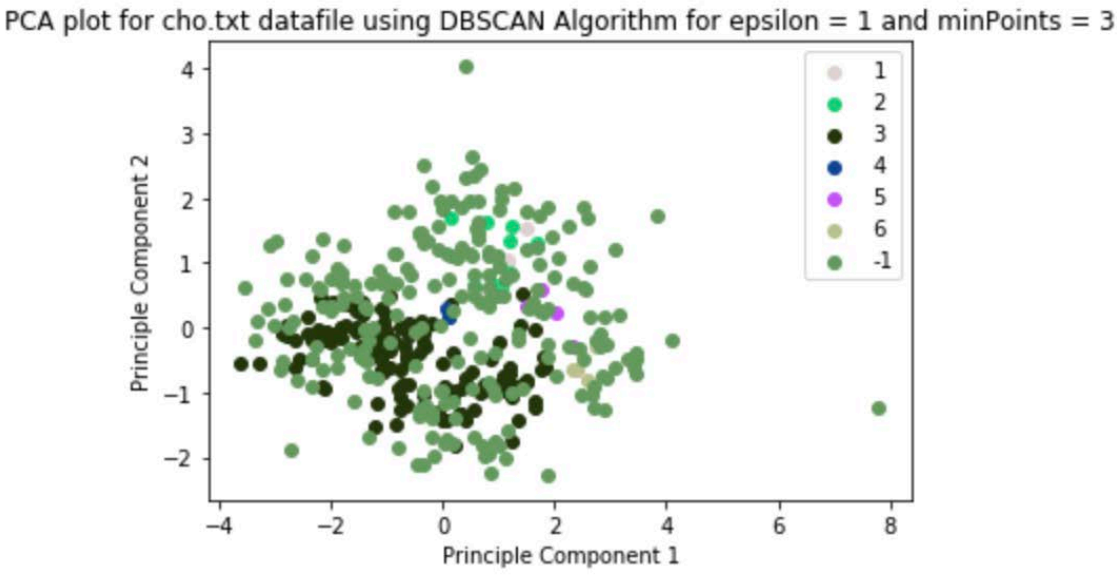
- As opposed to k-means, DBSCAN does not require one to specify the number of clusters.
- It can handle clusters of different shapes and sizes. It can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by a different cluster.
- DBSCAN is resistant to noise.
- It is insensitive to the ordering of the points in the database.

Disadvantages:

- Depending on the order the data is processed, border points that are reachable from more than one cluster can be part of either cluster. So DBSCAN is not deterministic.
- Choosing a meaningful distance threshold ε is difficult. So DBSCAN is sensitive to parameters.
- Because the minPts-ε combination cannot be chosen appropriately for all clusters, it cannot handle varying densities.
- It uses Euclidean distance to calculate the distance between two points. But for high dimensionality data, Euclidean distance is considered useless due to Curse of dimensionality.

Plots and Results:

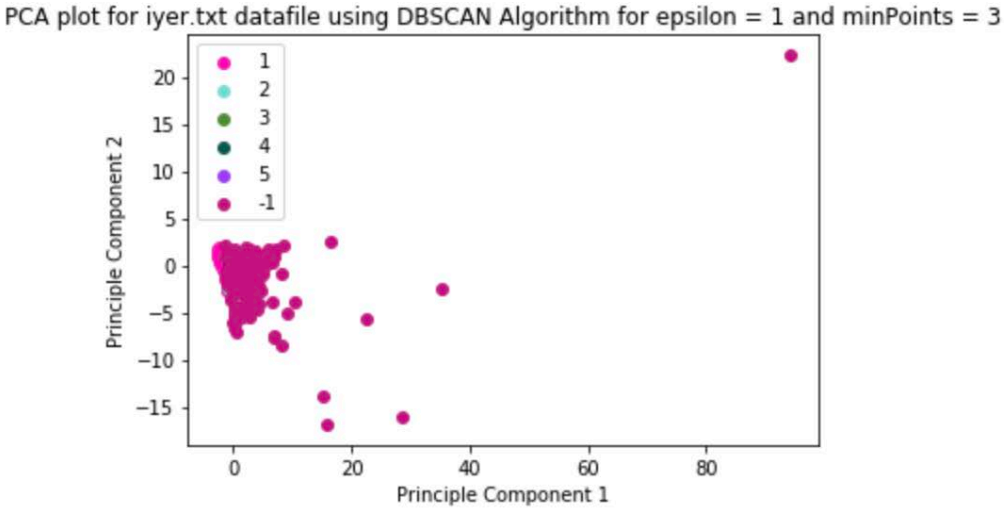
Cho.txt:



The Jaccard and Rand Coefficients for the above graphs and input parameters:

```
The Jaccard Cofficient for DBSCAN Algorithm for  clusters is: 0.20445756872671025
The Rand Index for DBSCAN Algorithm for  clusters is: 0.5443501838975543
```

Iyer.txt:




```
The Jaccard Cofficient for DBSCAN Algorithm for  clusters is: 0.2812609022614682
The Rand Index for DBSCAN Algorithm for  clusters is: 0.6531544508004444
```

Result Evaluations:

- Clusters formed are of different shapes and sizes which is helpful when the data is not regular.
- DBSCAN labels all the points as noise or core depending on the number of points reachable within the epsilon.
- As we can choose the minimum number of points required to form a dense region, we can decide the density of the clusters.

Algorithm 4: SPECTRAL CLUSTERING

In spectral clustering, the affinity between the points determines what points fall under which cluster. Spectral clustering is a technique that follows the approach- if two points are connected or immediately next to each other they are put in the same cluster. The data points are treated as nodes of a graph. The nodes are then mapped to a low-dimensional space so that they can be easily segregated to form clusters. No assumption is made about the shape/form of the clusters. The main idea is used to identify communities of nodes in a graph based on the edges connecting them.

Algorithm:

The following steps are followed to implement spectral clustering:

- Compute similarity, degree and Laplacian matrix
- Calculation of Eigenvalues and Eigenvectors.
- Identifying the number of clusters using Eigen gap.
- Performing K-Means and labelling the data points.

Compute similarity, degree and Laplacian matrix:

An undirected graph is constructed from the n observations of the data, where the vertices of the graph corresponds to the number of observations. A similarity matrix W which is a NxN matrix where N is the number of samples is constructed. The cells are filled with the Gaussian kernel distance (using the sigma value provided by the user) between each pair of points. After the similarity matrix is calculated, degree matrix is calculated. For each row of the degree matrix, cells along the diagonal are filled by summing all the elements of the corresponding row. Now Laplacian matrix is calculated by subtracting the similarity matrix from the degree matrix.

Calculation of Eigenvalues and Eigenvectors.

Calculate the Eigenvalues and Eigenvectors of the Laplacian matrix calculated in the previous step.

Identifying the number of clusters using Eigen gap.

The Eigen gap is calculated from the Laplacian matrix. It indicates the number of clusters.

Performing K-Means and labelling the data points.

The vectors associated with the eigenvalues contain information on how to segment the nodes. Finally, K-Means can be performed on those vectors in order to get the labels for the nodes.

Implementation:

- We have loaded the file using np.loadtxt function and removed the first two columns.
- We have given the option to enter sigma.
- Initialize similarity matrix(W) by N*N where N is the number of rows of the dataset.
- The cells of the W matrix are filled with the Gaussian kernel distance between each pair of points.
- The cells of the degree matrix(D) are calculated by placing on the diagonals the sum of all the elements in a single row.
- Laplacian matrix(L) is the difference of the Degree and the W matrix.
- We have calculated Eigen values and Eigen vectors from the eigh function of linalg from numPy. This returns us the sorted Eigen values.

- We have calculated the eigengap to decide the number of clusters. The distance between all the adjacent values in Eigen values list is calculated. We are considering the index+1 of the maximum of these distances as the eigengap.
- We are giving the option to the user to enter the number of centroids(K).
- From eigenvectors, we are taking all the n rows and "k_gap" columns, where k_gap is the value of the eigengap just calculated. This is the input to k-means.
- We are giving choice to either input the initial cluster centre indices or we are selecting the first K cluster centers as per the user's choice.
- We are importing KMeans from SpectralClustering of sklearn.cluster.
- The input to the KMeans function is the number of centroids, the initial centroid values input by the user and N* k_gap matrix obtained from eigen vectors.
- We have calculated the unique clusters values for plotting purpose.
- Dimensionality Reduction of given dataset. For dimensionality reduction of given dataset to 2 dimensions we use sklearn.decomposition PCA library function. We passed the normalized data into the PCA function and get the new factors.
- Plot the new factors on a matplotlib plot and color the point according to the cluster the gene id belongs to.
- Then find the Rand and Jaccard Coefficient. The steps for it is same as in the other algorithms explained above.

Advantages:

- Spectral clustering does not make assumptions on the shape of the clusters which helps
    - in creating more accurate clusters.
- It gives good clustering results and it is easy to implement.
- Even for sparse data sets it is fast.

Disadvantages:

- Clusters will not always be the same. They may vary depending on the choice of initial centroids. This is because we will use k-means clustering in the final step of spectral clustering.
- They are computationally expensive for large datasets.As eigenvalues and eigenvectors need to be computed and clustering has to be performed on these vectors, time complexity may increase.

Applications:

- To identify objects in satellite images.

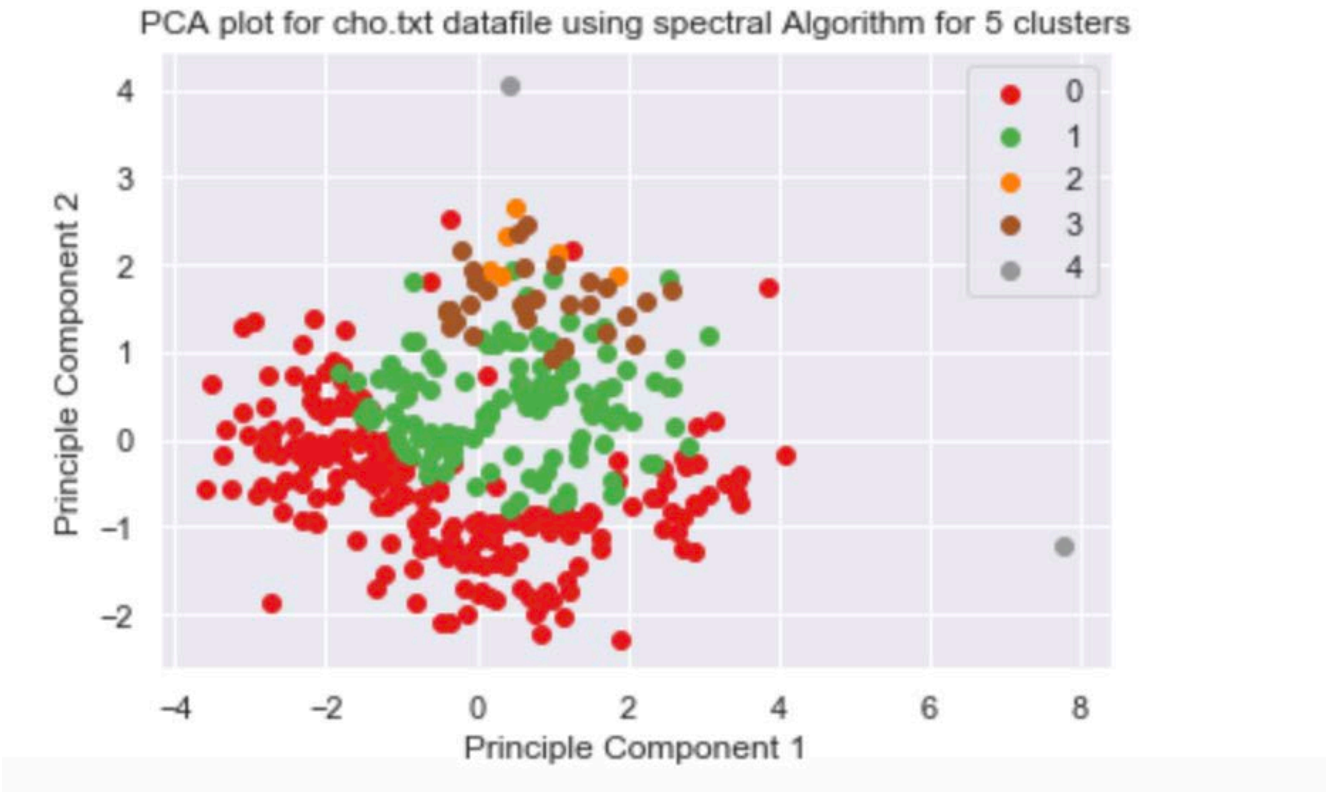
Plots and Results:

Input 1(cho.txt):

- File Name: cho.txt
- Enter sigma: 2
- Enter the total number of centroids:5
- Enter initial centers indices: 1,3,5,7,9

Output 1(Cho.txt):

```
The file name is: cho.txt
The number of centroids i.e. cluster centers are: 5
Initial centroids are:
[[-0.05089866  0.00259719 -0.00241473]
 [-0.05089866  0.002597   -0.0022857 ]
 [-0.05089866  0.00259574 -0.00194073]
 [-0.05089866  0.00259604 -0.00199656]
 [-0.05089866  0.00259631 -0.00100372]]
New updated centroids after k means application:
[[-0.05089866  0.00259254 -0.00301522]
 [-0.05089866  0.00259584 -0.00249512]
 [-0.05089866  0.00259697  0.00272013]
 [-0.05089866  0.00259617 -0.00121567]
 [-0.05089866 -0.49804938  0.49925753]]
The RAND coefficient is: 0.576861123788558
The JACCARD coefficient is: 0.22486967640405234
```

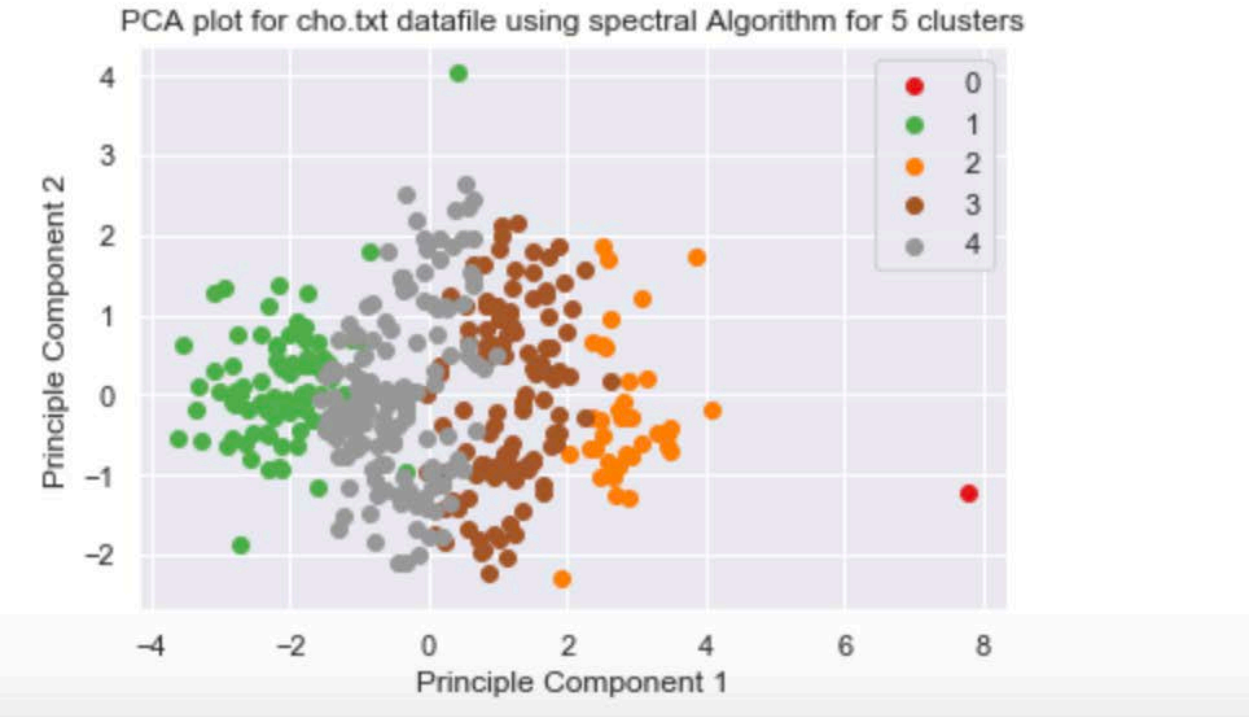


Input 2(cho.txt):

- File Name: cho.txt
- Enter sigma: 10
- Enter the total number of centroids:5
- Enter initial centers indices: 1,3,5,7,9

Output 2:

```
The file name is: cho.txt
The number of centroids i.e. cluster centers are: 5
Initial centroids are:
[[-0.05089866  0.00227186]
 [-0.05089866  0.00255063]
 [-0.05089866  0.00230806]
 [-0.05089866  0.00242464]
 [-0.05089866  0.00247499]]
New updated centroids after k means application:
[[-5.08986599e-02 -9.98572818e-01]
 [-5.08986599e-02  3.61129195e-03]
 [-5.08986599e-02  9.50927587e-04]
 [-5.08986599e-02  2.09013512e-03]
 [-5.08986599e-02  2.84703937e-03]]
The RAND coefficient is: 0.6831727026228892
The JACCARD coefficient is: 0.24792888095845017
```

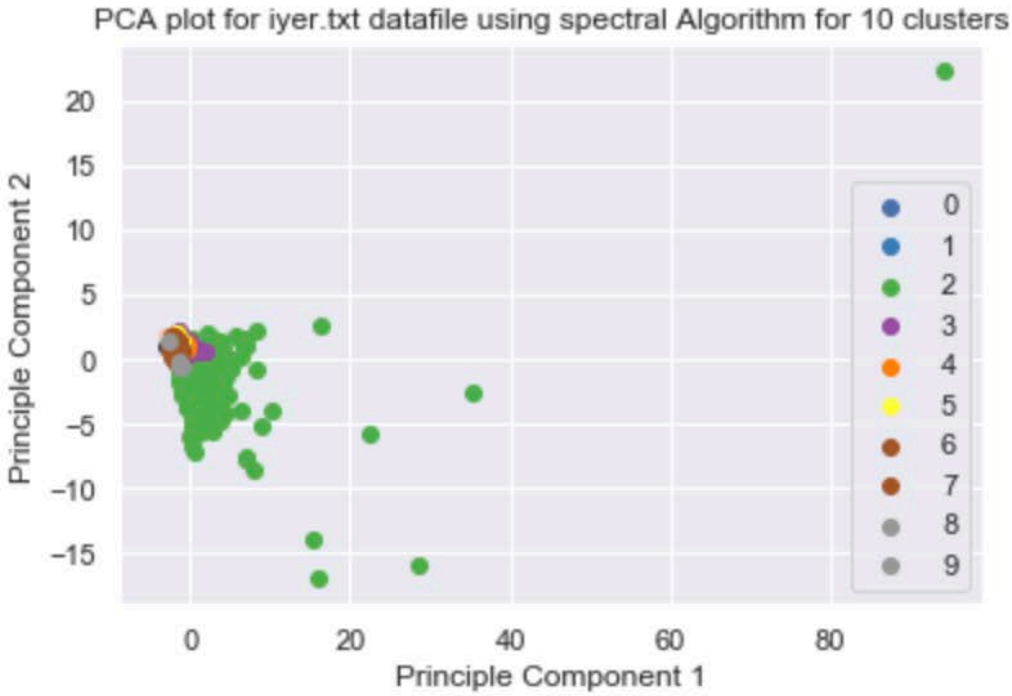


Input 3(iyer.txt):

- File Name: iyer.txt
- Enter sigma: 2
- Enter the total number of centroids:10
- Enter initial centers indices: 1,3,5,7,9,11,13,15,17,19

Output 1:

```
The file name is: iyer.txt
The number of centroids i.e. cluster centers are: 10
Initial centroids are:
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  3.22515762e-03
  -9.18364554e-03  3.39267517e-03]
 [-1.91895526e-17 -4.61555386e-19  3.39782261e-17 ...  1.62043408e-03
  -1.71114080e-03 -4.85387697e-03]
 [-3.24109658e-17 -4.00525375e-19  1.13876595e-19 ...  1.03472604e-03
  -4.59336107e-04 -6.36180403e-03]
 ...
 [-1.64312757e-17 -9.03575779e-21  7.75612025e-18 ...  2.16780519e-03
  -6.12285238e-03  6.65821129e-04]
 [ 1.01862658e-16  7.40660055e-17  1.89576610e-16 ...  1.77637132e-03
  -3.75052666e-03 -1.97540484e-03]
 [-1.74812038e-18 -1.84223765e-19 -2.96978418e-18 ...  3.22212975e-03
  -6.95646539e-03  4.49132024e-03]]
New updated centroids after k means application:
[[-4.33680869e-19  0.00000000e+00 -1.51788304e-18 ...  4.43084427e-03
  -1.18450581e-02  1.41071817e-02]
 [-3.03576608e-18  0.00000000e+00  4.33680869e-19 ...  1.15384377e-03
   3.98533297e-03  3.25152051e-03]
 [-6.34822673e-03  5.11020611e-03 -5.26314760e-03 ...  8.79941249e-04
   3.26964503e-04  5.14854035e-04]
 ...
 [-1.30104261e-18  3.25260652e-18  3.68628739e-18 ...  1.56004582e-03
  -3.54587828e-03 -2.36726902e-03]
 [-1.73472348e-18  2.16840434e-19  6.50521303e-19 ... -1.72640816e-01
   4.66194391e-02  2.00210952e-02]
 [-4.33680869e-19  0.00000000e+00 -1.30104261e-18 ...  1.14718001e-02
  -3.46343612e-02  9.88812623e-01]]
The RAND coefficient is: 0.73873223364972
The JACCARD coefficient is: 0.269021824462239
```

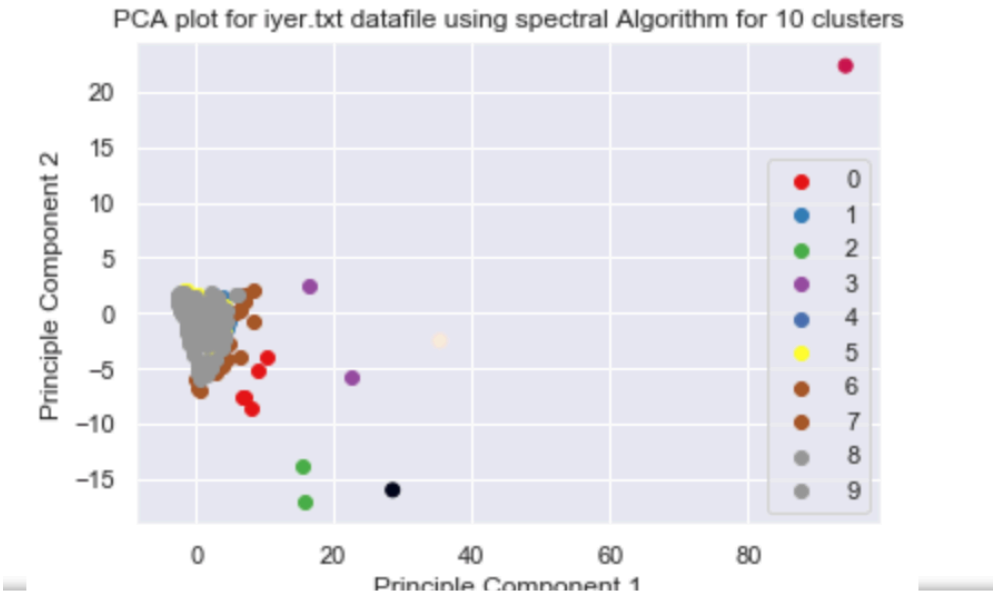


Input 4(iyer.txt):

- File Name: iyer.txt
- Enter sigma: 10
- Enter the total number of centroids:10
- Enter initial centers indices: 1,3,5,7,9,11,13,15,17,19

Output 2:

```
The RAND coefficient is: 0.5783814522857282
The JACCARD coefficient is: 0.23361918298231177
```



RESULT EVALUATION:

- In cho.txt, as we are increasing the sigma value from 2 to 10, we are getting proper clustering results. Rand and Jaccard value is increasing.
- In cho.txt, when we increased the sigma value to 50, there is no noticeable change in the values of Rand and Jaccard. So,10 is a better sigma value as per my experiment results.
- But the same trend is not being observed when we ran code for iyer.txt. Rand and Jaccard did not increase when we changed sigma value from 2 to 10. Rand and Jaccard value is better for sigma value 2.

Applications:

- Image segmentation
- Power load time-series clustering
- Document clustering

Algorithm 5: Gaussian Mixture Model Clustering

- A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by $k \in \{1,\ldots,K\}$, where K is the number of clusters of our dataset. It uses the soft clustering approach for distributing the points in different clusters.Each Gaussian ***k*** in the mixture is comprised of the following parameters:
    - Mean μ – centre.
    - Covariance Σ - width.
    - Mixing probability π,
      The π is dependent on how big or small the Gaussian function will be.

- **Expectation maximization** is the technique used to estimate the mixture model's parameters. It is a numerical technique for maximum likelihood estimation.
- It is an iterative algorithm that makes sure that the maximum likelihood of the data increases with each subsequent iteration and it is guaranteed to approach a local maximum or saddle point.

Expectation maximization consists of two steps:

- Expectation step or E step: For each data point, this step calculates the probability(expectation) of the component assignments, for the given parameters μ, Σ and π. As we know μ, Σ and π, making inferences about the probability of the datapoint is easy. It creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters.
- Maximization step or M step: This step consists of maximizing the expectations calculated in the E step with respect to the parameters. This step consists of updating the values μ, Σ and π. Knowing the component assignment for each data point, simplifies solving μ, Σ and π. This step maximizes the expected complete log-likelihood.
- The entire iterative process is repeated until the algorithm converges, giving a maximum likelihood estimate or a threshold value.

Advantages:

- Gives probabilistic cluster assignments. The probability of belonging to each cluster is calculated and a classification is usually achieved by assigning each observation to the most likely cluster. Performs soft clustering.
- Can be used for non-spherical clusters. More flexible in terms of cluster covariance and varying sizes.

Disadvantages:

- Choosing the number of components is an issue. If we choose too many components it will cause over-fitting. On the other hand, if we choose few components. It makes the model less flexible.
- If we do not initialize mean and sigma values properly. They will not reach maximum likelihood and might only reach local maxima.

Implementation:

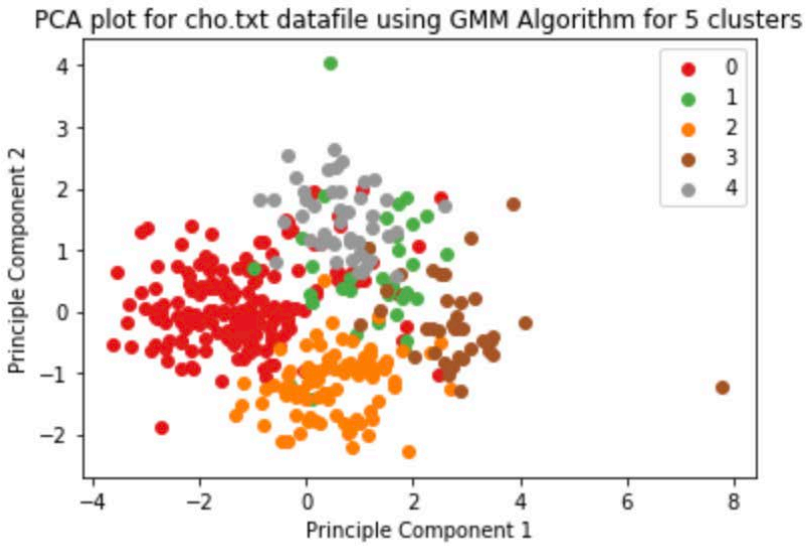
- We are reading the file and converting it to numpy matrix using genfromtxt function of numpy. From this data, we have removed the first two columns.
- For Standardization purposes we remove the columns with zero variance. This step may not be required for the given datasets but is a standard across industry.
- Options are given to enter the parameters: mu, pi and sigma or initialize using kmeans.
- We have given the options to enter the number of clusters, max iterations, threshold value and smoothing value.
- We are running K-means to initialize mu, pi and sigma.
- Now we go to E step We have calculated the probability(expectation) of the data point belonging to a particular gaussian using mu, pi and sigma by using the below formula:

  probability = 1/pow((2*math.pi), -1/2) * pow(abs(np.linalg.det(Si)), -1/2) * p
- Using the probability obtained in the previous step, we have computed a better estimate for the mu, pi and sigma parameters.(M-step).
- We are calculating the log-likelihood with each iteration. We will repeat these steps until convergence.
- We find the cluster assignment by using the final weights. The cluster with the maximum weight for a gene id gets the gene id assigned in the cluster.
- Perform dimensionality reduction using PCA and plot the points as per cluster assignment.
- Finally, we calculate Rand and Jaccard Coefficient, using the same steps as shown in above algorithms.

Results:

Input 1(cho.txt):

- File Name: cho.txt
- Number of iterations: 100
- Number of clusters : 5
- Threshold value: 1e-08
- Smoothing Value: 0.0001

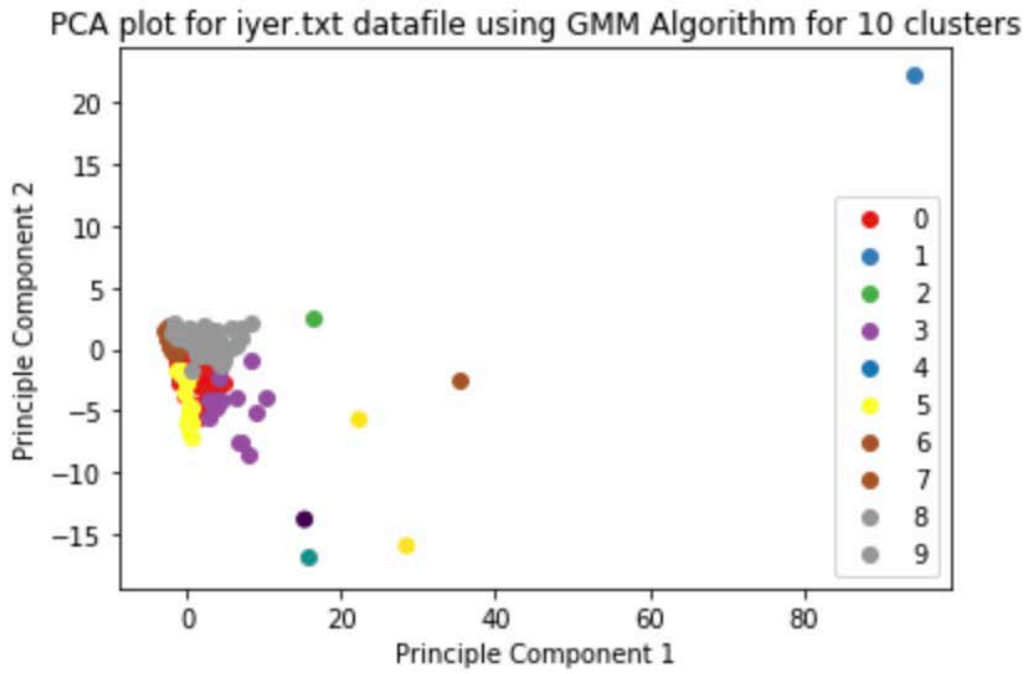


The Rand and Jaccard Cofficients for the above shown graph are:

```
The Jaccard Cofficient for GMM Algorithm(cho.txt) for 5 clusters is: 0.3743696748391584
The Rand Index for GMM Algorithm(cho.txt) for 5 clusters is: 0.7585170071679777
```

Input 1(iyer.txt):

- File Name: iyer.txt
- Number of iterations: 100
- Number of clusters: 10
- Threshold value: 1e-08
- Smoothing Value: 0.0001




```
The Jaccard Cofficient for GMM Algorithm(iyer.txt) for 10 clusters is: 0.34683805238694587
The Rand Index for GMM Algorithm(iyer.txt) for 10 clusters is: 0.753427937550741
```

Result Evaluation:

- We have implemented k-means through which we are extracting the values for mu,pi and sigma. This enhances the results by providing a refined values for the above parameters.
- The results hence obtained are close to the close to the library implementation

References:

- https://link.springer.com/chapter/10.1007/978-3-319-24211-8_6
- https://brilliant.org/wiki/gaussian-mixture-model/
- https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95
- https://en.wikipedia.org/wiki/DBSCAN
- https://link.springer.com/chapter/10.1007/978-3-319-24211-8_6
- https://towardsdatascience.com/spectral-clustering-aba2640c0d5b
- Data Mining Concepts and Techniques – Jiawei Han, Micheline Kamber and Jian Pei – Morgan Kaufmann
- https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique
- Introduction to Data Mining. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison Wesley.
- http://santini.se/teaching/ml/2016/Lect_10/10c_UnsupervisedMethods.pdf
- https://en.wikipedia.org/wiki/K-means_clustering
- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.pyplot.html