CSE 601 : DATA MINING

CLUSTERING ALGORITHMS

TEAM 27

Name	Email	UB ID
Akash Yeleswarapu	akashyel@buffalo.edu	50207826
Maheedhara Achalla	maheedha@buffalo.edu	50207395

1 K-MEANS CLUSTERING

OBJECTIVE:

Implementing K-Means clustering on the cho and iyer data sets, validating the clusters by computing Jaccard coefficient or Rand Index using given ground truth values and externally known results and plotting the clusters by Principal Component Analysis.(PCA).

DEFINITION:

K-Means clustering aims to partition “n” observations in to “k” clusters in such a way that each observation in the cluster is closer to the center of the cluster than to center of the any other cluster. “K” represents the number of clusters which is usually specified.

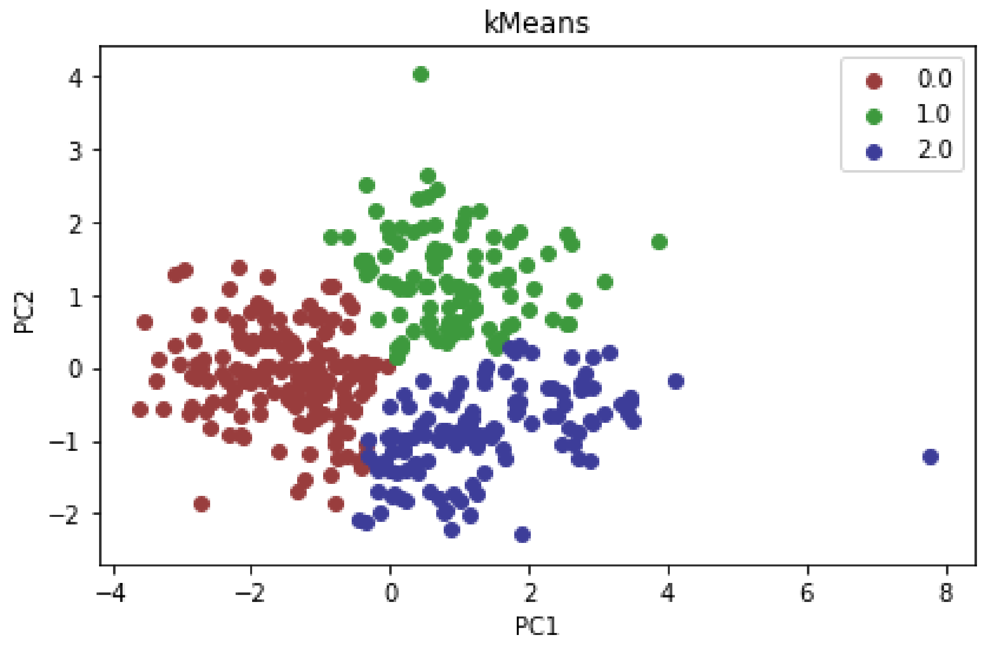
ALGORITHM:

1) Partition of the observation set into k (initially specified) clusters.
2) Specify the initial cluster centers (centroids).
3) Compute the distance between each observation and all the centroids. Assign the Observation to the centroid with the minimum distance.
4) Compute the centroid of each clustered observations which become the new cluster Centroids.
5) Repeat from Step 3 until it converges. (until previous and current assignment of centroids are same).

RESULTS:

We have implemented K-Means clustering and run them on the two sample datasets ‘cho.txt’ and ‘iyer.txt’.
Also, the results are visualized by comparing with PCA by plotting them on the graph.
The results are given below:

cho.txt
k = 3:
Iterations before converging 17
Jaccard-Coefficient 0.4088574886337618
Rand Index 0.7574028832988805



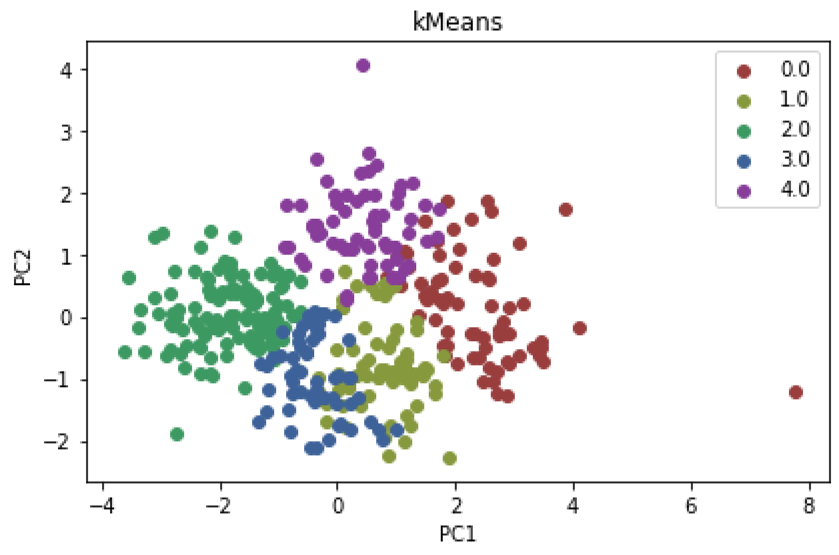
cho.txt

k = 5:

Iterations before converging 7

Jaccard-Coefficient 0.4064635895705004

Rand Index 0.8111627157776048



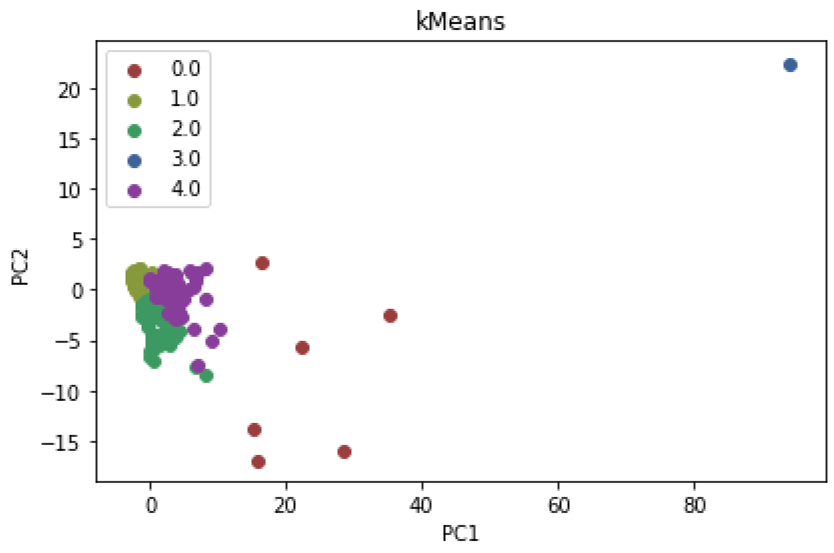
iyer.txt

k = 5:

Iterations before converging 24

Jaccard-Coefficient 0.2755972252169193

Rand Index 0.6354881794611824

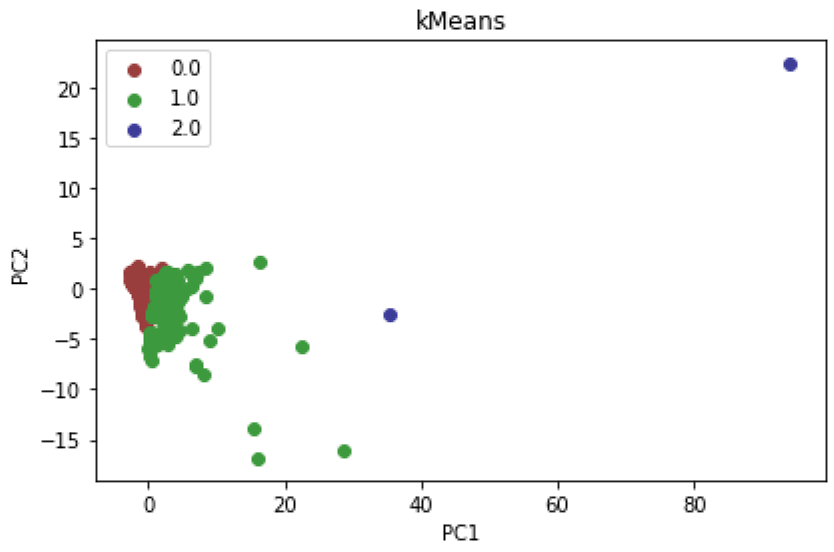


K=3

Iterations before converging 23

Jaccard-Coefficient 0.2199408701534227

Rand Index 0.4995304707638549



Increasing the k value improves the jaccard coefficient and data points are better clustered.

OUR OBSERVATIONS

Choice of K

K-Means algorithm is sensitive to the initial parameters. We found out that, if initially, k is not specified, we can get the optimal value of k by running the algorithm multiple times keeping the same initial centroids and checking the Sum of squared error values(SSE). After one value of k, there will be minimal change in the SSE value. We can select that k value as "K optimal".

Choice of Initial Centroids

For this project, we chose the initial centroids randomly. We picked up the random datapoints as our initial centroids. We chose 5 datapoints [5,25,32,100,132] as our initial centroids and run the algorithm. We found that the best way to choose the initial centroids is to find the k points which are most far apart and make them the initial centers. This reduces the time taken for the convergence. If we randomly choose K points as centers and there are chances that chosen points are too close to each other. This increases the time to converge and increases the running time of the algorithm.

Ex:

We chose points belonging to the same cluster (based on the ground truth) as initial centroids and points belonging to different clusters (based on the ground truth) for testing this. We observed that choosing initial centers from the same cluster takes more time to converge than the initial centroids from different clusters.

Also by studying research papers on this topic, we found that choosing initial centroids based on the weighted method (K++ means) gives better method. But this is not always true. Sometimes random initialization works better than the K++ means initialization.

PROS:

- Easiest to implement
- Can be used as a preprocessing step for other clustering algorithms
- Suited for elliptical shaped clusters
- If each cluster has equal density and distribution, K means is suitable

CONS:

- If k is not specified, finding optimal k value takes significantly more time on the large datasets
- Initial selection of centroids decides the time to converge.

- There are chances that empty clusters might appear
- k-means does not work well when there are clusters of different shapes, densities and sizes.
- k-means is not able to identify outliers as it assigns each datapoint to one of the cluster (which contributes to most of the SSE)

2 HIERARCHICAL AGGLOMERATIVE CLUSTERING

OBJECTIVE:

Implementing hierarchical agglomerative clustering with Min Link on the given datasets, validating the clusters by computing Jaccard coefficient or Rand Index using given ground truth values and externally known results and plotting the clusters by Principal Component Analysis. (PCA).

DEFINITION:

Hierarchical clustering takes agglomerative approach where each point in the dataset is considered as an individual cluster and continues in merging the clusters based on different approaches. The different types of approaches in merging two clusters into one are:
1. MIN or Single Link: We consider the minimum distance between all the points in the two clusters in computing the inter-cluster distance.
2. MAX or Complete Link: We consider the maximum distance between all the points in the two clusters in computing the inter-cluster distance.
3. Group Average or Average Link: We consider the average distance between all the points in the two clusters in computing the inter-cluster distance.
4. Centroid Distance: We consider the distance between the centers of the two clusters in computing the inter-cluster distance.

Each approach has its own pros and cons. We'll be considering a approach based on our dataset. Each dataset might need a different approach to get desired results.

ALGORITHM DESCRIPTION:

Here we consider MIN or Single Link for calculating the inter-cluster distances for our clustering. The algorithm is:

1. Consider each point in the dataset as a cluster initially
2. Calculate the distance matrix of all the points
3. Find the two closest clusters and merge them. Here we use MIN as our criterion in finding the clusters that are to be merged.
4. We update the distance matrix by the intersection of these two clusters
5. Repeat from Step 1 till we're left with a single cluster or till the number of clusters become some 'k' value

DENDROGRAM:

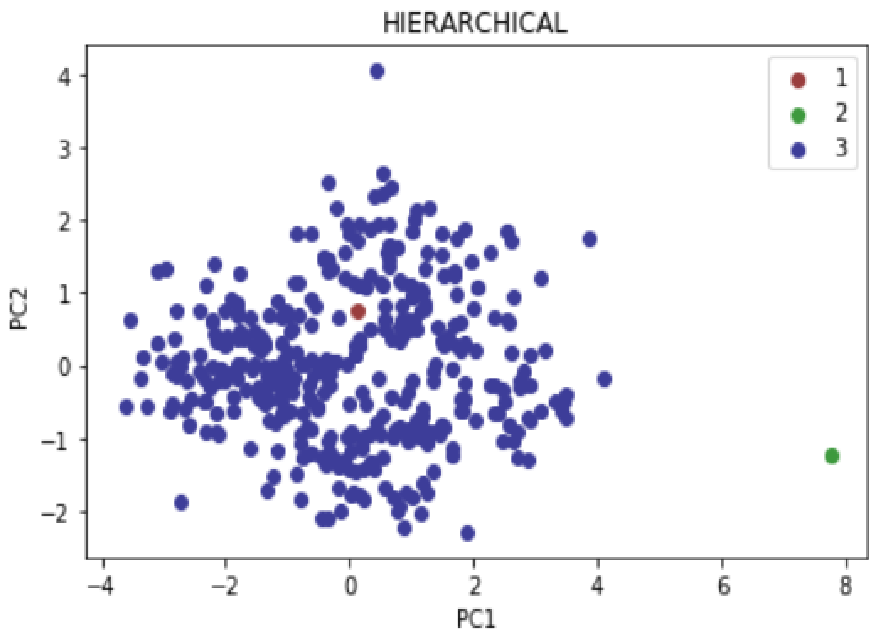
Dendrogram is a tree that can be used to visualize the merging of clusters. Initially each point is considered as a cluster. So, they're represented individually. We keep merging two clusters in each iteration until only one cluster remains.
We can cut the dendrogram at any desired level to get the separate clusters.
For example, cutting the dendrogram at level 1 gives us 2 main clusters and level 2 gives us 3 main clusters and so on.

RESULTS:

We've implemented Hierarchical clustering and run them on the two sample datasets 'cho.txt' and 'iyer.txt'.
Also, the results are visualized by comparing with PCA by plotting them on the graph.
The results are given below:

For cho.txt with k = 3:
JaccardCo-efficient:0.22933831189036352
RandIndex: 0.2353485999624151

The graph is:

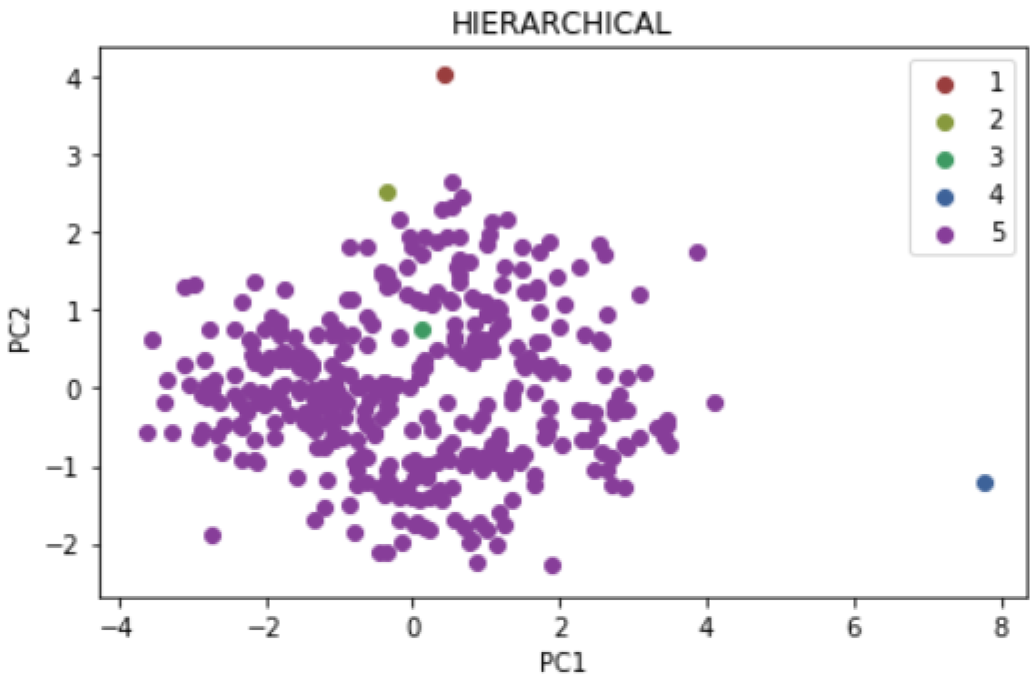


For cho.txt with k = 5:

JaccardCo-efficient:0.22839497757358454

RandIndex: 0.24027490670890495

The graph is:

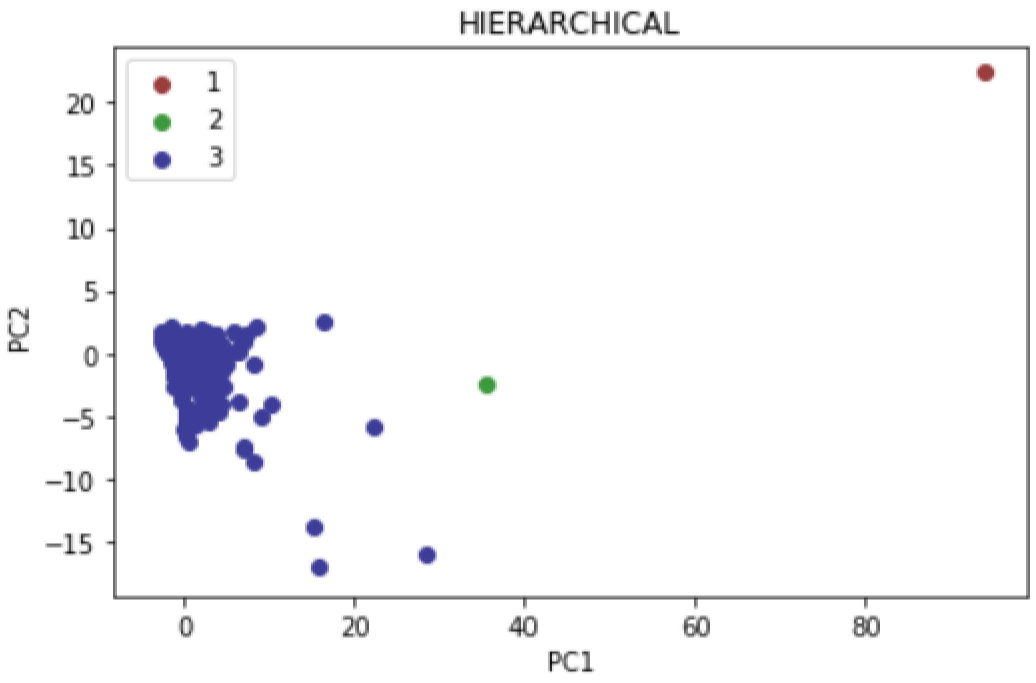


For iyer.txt with k = 3:

JaccardCo-efficient:0.1559309385706048

RandIndex: 0.1621204015129691

The graph is:

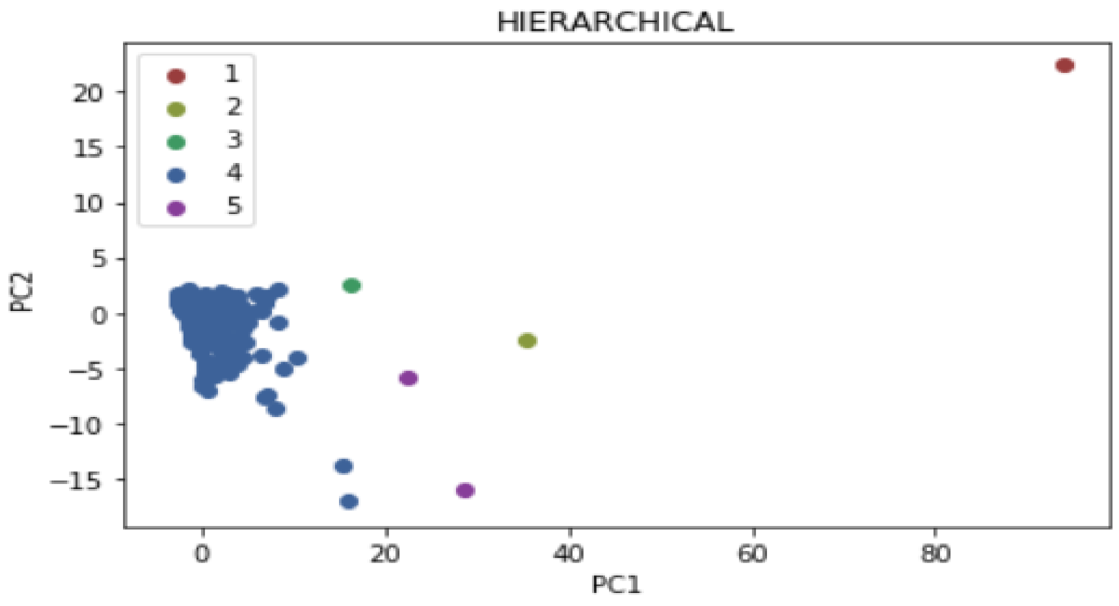


For iyer.txt with k = 5:

JaccardCo-efficient:0.15647222380925174

RandIndex: 0.17144364339722173

The graph is:



RESULT EVALUATION:

As we can see from the above graphs, the natural tendency of MIN link of favoring elliptical shapes caused a trouble in merging most of the clusters into a single cluster and its limitation of being susceptible to noise and outliers made it consider some noise as separate clusters.
This is also reflected in the low jaccard coefficient and rand index values. Both cho and iyer datasets have shown poor performance with this algorithm but cho was a bit better than iyer. This could be because of more outliers in iyer dataset. It could've had better results with MAX Link since it handles outliers better than MIN link

PROS:

1. The main advantage of using hierarchical clustering is with its ability to form clusters without having to specify the number of clusters to be formed. Any desired number of clusters can be derived by cutting the dendrogram a particular level.
2. The advantage of MIN comes into picture when the data is aligned in elliptical shapes as MIN link favors elliptical shapes.

CONS:

1. One of the main disadvantages of using hierarchical clustering is with its running time. Most of the approaches take O(N3) time which could be a concern for huge datasets.
2. The main disadvantage of using MIN Link is its inefficiency in handling noise or outliers as seen above.

3 DBSCAN CLUSTERING

OBJECTIVE:

Implementing DBSCAN clustering on the cho and iyer data sets, validating the clusters by computing Jaccard coefficient or Rand Index using given ground truth values and externally known results and plotting the clusters by Principal Component Analysis. (PCA).

DEFINITION:

DBSCAN is density based clustering technique. In this method, cluster is viewed as set of densely connected points. They are dense regions in the data space, separated by regions of lower object density. It is characterized by epsilon neighborhood and MinPts. Epsilon neighborhood indicates the all the points within the radius of epsilon from the current data point. Epsilon neighborhood of the datapoints contains at least MinPts number of datapoints. Idea is to cluster the datapoints based on epsilon and MinPts.

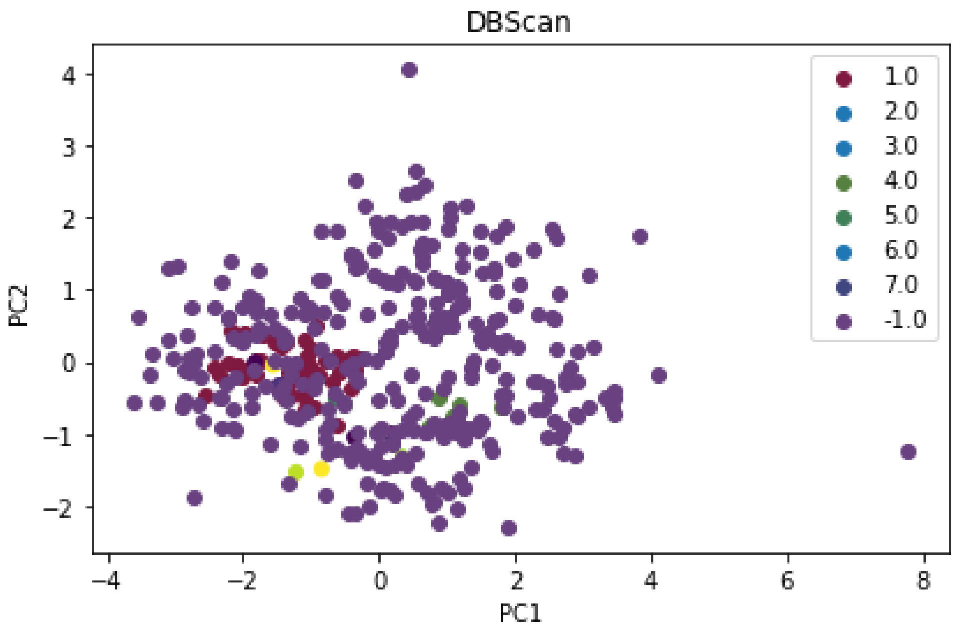
ALGORITHM:

1. For each data point in the dataset, find the list of the nearest neighbors separated by epsilon distance. Mark the point as visited.
2. Check whether the number of nearest neighbors is greater than MinPts.
3. If No, add the point to the Noise.
4. If Yes, add the point to a cluster and repeat the process for its nearest neighbors separated by epsilon.
5. Do it iteratively until all the points are marked as visited.

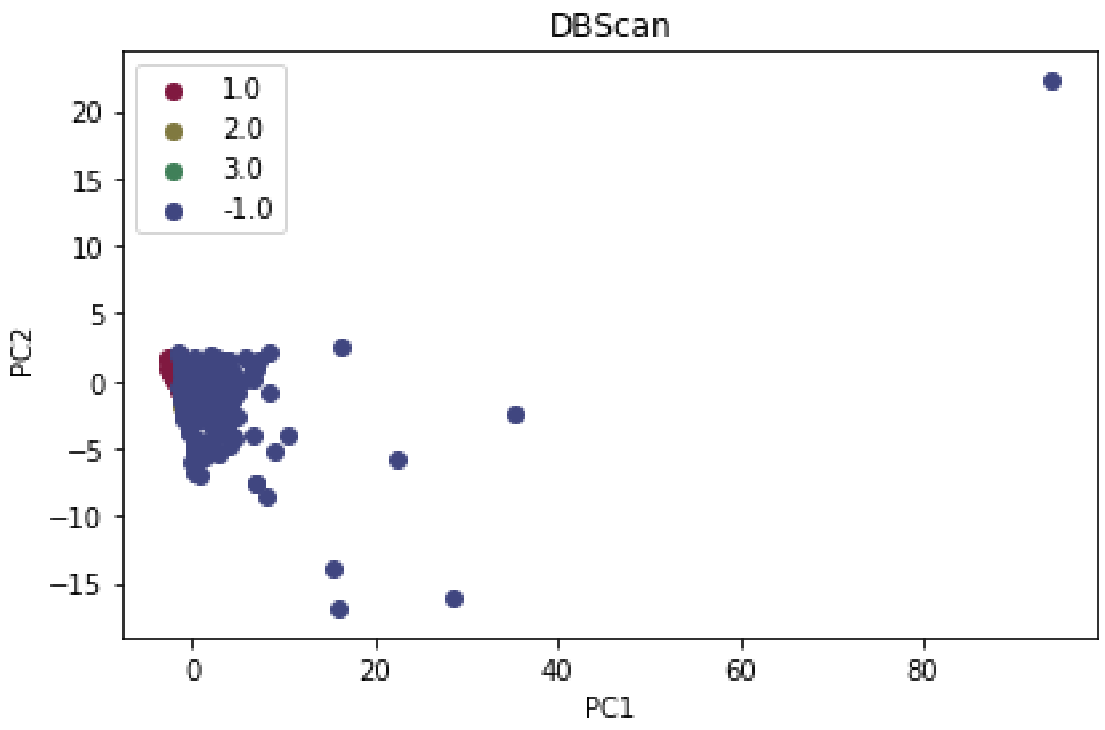
RESULT:

We've implemented DBSCAN clustering and run them on the two sample datasets 'cho.txt' and 'iyer.txt'.
Also, the results are visualized by comparing with PCA by plotting them on the graph.
The results are given below :

Cho.txt
esp = 0.9, MinPts =4
Jaccard-Coffecient 0.20265509761388287
RandIndex 0.3832451877902763



Iyer.txt
esp = 0.9 MinPts =4
Jaccard-Coffecient 0.28322121478015017
RandIndex 0.6424244918421633



Result Evaluation
Density of the data distribution varies in the iyer data set. Algorithm is not so effective in clustering in this case. More outliers are observed in the scatter plot of the iyer data set. Some of the clusters might have included in the outliers. It depends on the way data has been chosen for clustering. It was very tough to obtain the optimal values of esp and MinPts. It is equally bad on the cho data set and outputted many data points as outliers. Jaccard coefficients of both the data sets are very poor (around 0.2)

PROS

- Algorithm can handle non-spherical clusters. It can handle clusters of different shapes and sizes.
- DBSCAN can effectively identify the outliers. It is resistant to noise.

CONS

- Algorithm suffers from parameter sensitivity. It is very tough to obtain the optimal values of epsilon and MinPts.
- If the density of points in the distribution varies, algorithm cannot handle.

4 K-MEANS CLUSTERING BY MAP-REDUCE

OBJECTIVE:

Implementing K-Means clustering using Map-Reduce algorithm on a single node Hadoop cluster on the given datasets, validating the clusters by computing Jaccard coefficient or Rand Index using given ground truth values and externally known results, comparing the results with non-parallel K-Means and plotting the clusters by Principal Component Analysis. (PCA).

DEFINITION:

K-Means clustering aims to partition “n” observations in to “k” clusters in such a way that each observation in the cluster is closer to the center of the cluster than to center of the any other cluster. “K” represents the number of clusters which is usually specified.

ALGORITHM:

We’ll be using MapReduce algorithm to perform this task.

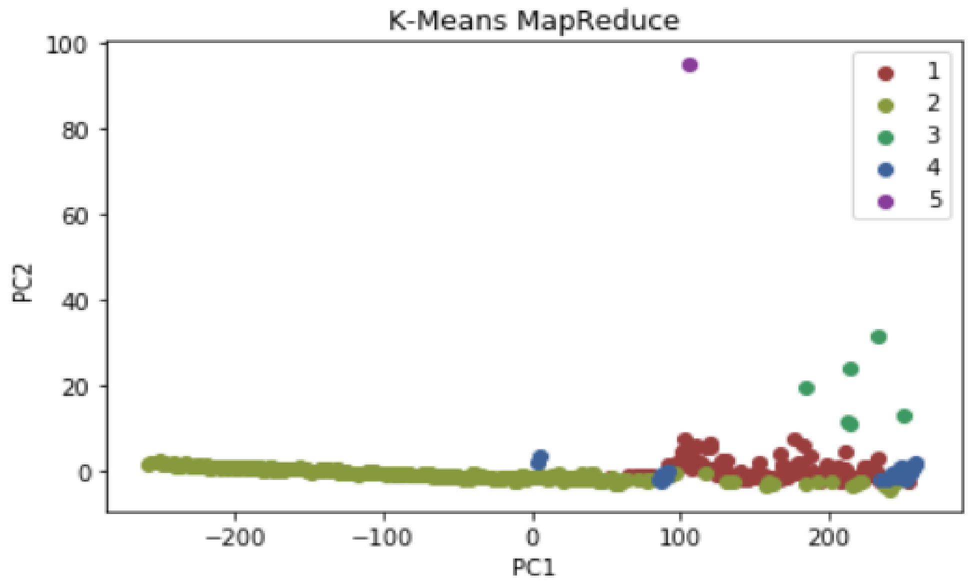
1. Partition of the observation set into k (initially specified) clusters.
2. Specify the initial cluster centers (centroids).
3. In the mapper, assign each point to its closest cluster
4. In the reducer, update the cluster centers by calculating the mean of all the points in the cluster.
5. Repeat step 3 and step 4 until the centroids converge.

RESULTS:

We’ve implemented K-Means by MapReduce and run them on the two sample datasets ‘cho.txt’ and ‘iyer.txt’.
Also, the results are visualized by comparing with PCA by plotting them on the graph.
The results are given below:

For iyer.txt with k = 5:
JaccardCo-efficient:0.2656728439207627
RandIndex: 0.6043234102413493
Time taken: 30836ms

The graph is:



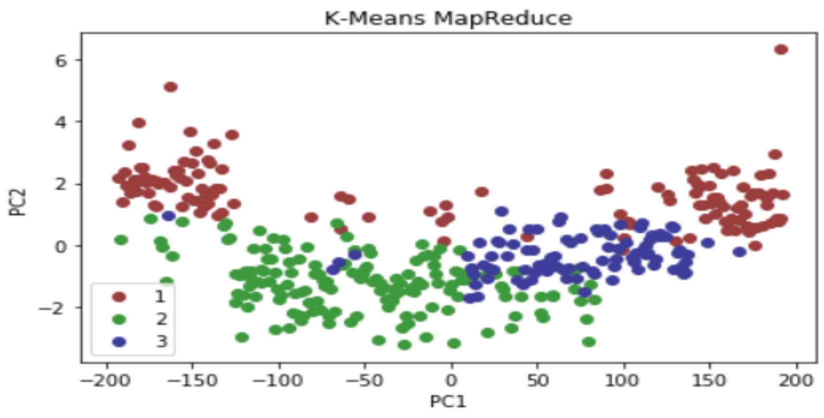
For cho.txt with k = 5:

JaccardCo-efficient:0.4292561158262606

RandIndex: 0.7698193240086982

Time taken: 29281ms

The graph is:



RESULT EVALUATION:

As we can see from the graphs, K-Means worked way better than Hierarchical Clustering for these datasets. Between cho and iyer datasets, cho showed a better jaccard coefficient and rand index values. This could be because of the presence of many outliers in iyer dataset.
Also, the number of iterations varied significantly based on the initialization of centroids.
Adding a combiner or a partitioner adds up the performance of the MapReduce algorithm.

K-MEANS PARALLEL VS NON-PARALLEL:

K-Means when run on Hadoop by MapReduce had significant performance in terms of running time when compared to non-parallel way for large datasets. However, when a small dataset was provided as an input for parallel process, it took way more time than non-parallel process. This is because of the time taken for initialization of Hadoop and configuration before the actual algorithm starts working. Also, since there are way more files to read and write in parallel process, that would consume some time as well. So, it's suggestible to use parallel way only for large datasets.

PROS:

1. Significant performance for big datasets
2. Easy to implement
3. Suited for elliptical shaped clusters

CONS:

1. Sensitive to initial parameters
2. Empty cluster might show up
3. Can't handle irregular shapes.

COMPARING THE ALGORITHMS:

It is very difficult to performance of K-means algorithm on two different data sets cho and iyer as we take two separate initial centroids for both the data sets. Random seeding plays a pivotal role as the algorithm is sensitive to initial parameters. Same is the case with the DBSCAN algorithm. Algorithm performed poorly on both the data sets. When the density is varying in the data set DBSCAN performs poorly.

For the given datasets cho and iyer, hierarchical agglomerative clustering showed a poor performance with MIN Link because of outliers and favoring elliptical shapes. K-Means MapReduce showed better performance over K-Means for big datasets but not for small datasets. And comparatively, cho had better results with K-Means than iyer. This could be because of the outliers in iyer dataset.

CONCLUSION:

There is no algorithm that can work optimally for every dataset. Each clustering algorithm has its own pros and cons. However, by understanding the dataset, we can choose the algorithm fits that well and use it to achieve desired results.