Clustering Analysis

Roll: 1607110
Link for colab:
https://colab.research.google.com/drive/1Ymep87QdmBQtDCW508AtVXQ6rE9HqWSn?usp=sharing

Introduction:
In this assignment, we are comparing the similarity and dissimilarity of different clustering algorithms. Mainly k means, k medoid, AGNES, Birch, DBSCAN and CHAMELEON algorithms are analyzed on various data sets.

Datasets:
Link for dataset1: https://www.kaggle.com/janithwanni/old-faithful

Link for dataset2:https://www.kaggle.com/heptapod/titanic

Here we are using four datasets which are denoted as dataset1 ,dataset2, dataset3 and dataset4

Dataset1: Old Faithful Dataset which has three features. The first one is id , the second is eruptions and the third is waiting. This dataset has total of 273 samples
Dataset2: titanic dataset which has total of 1309 samples and 28 columns.
Dataset3: This a randomly generated two-dimensional dataset that has 1000 samples.
Dataset4: This a synthetic dataset from sklearn dataset called make_moons.which also has 1000 sample data points.

Preprocessing of datasets:

To get Standardize features standards StandardScaler preprocessing is used in dataset1 and dataset2. Only eruption and waiting features are considered in dataset1 and only age and fare features are considered for dataset2.

K means algorithms parameters: source[7]
n_clusters=2: Number of clusters to generate. here we want two clusters for dataset1 and dataset2 so n_clusters=2 is used.
N_init=10: for each centroid seeds the k means algorithm will run 10 times
max_iter=300: The algorithm will iterate a maximum of 300 times for a single run
tol=1e-4: tol denotes tolerance.The difference in two consecutive iterations in the cluster centers

precompute_distances='auto': computes distance because 2*273<12 million for dataset1 and 2*1309<12 million in dataset2

verbose=0: the verbosity mode is disabled

init='random': initial centroids are chosen randomly from the datasets

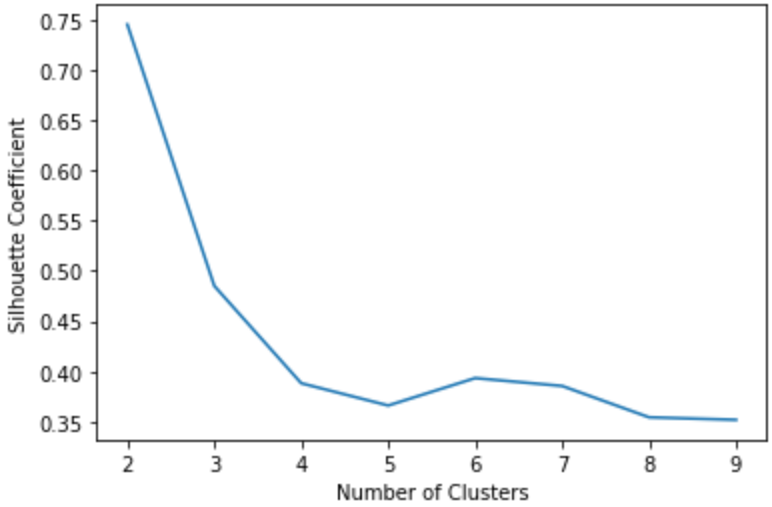
random_state=0: random seed 0 is used to produce the same result across different calls.

copy_x=True: The original data is not modified for dataset1 and dataset2.

n_jobs=None: The number of OpenMP threads that would be used for the estimation. All processors are used here.

algorithm='auto': which uses "elkan" variation which has well-defined clusters.

For k=2,3,4,,,,10 Silhouette Coefficient graph is



From this graph, we can see that with the increasing number of clusters the performance of k means algorithm decreases.The idea of this graph is generated from source [1]

Birch Algorithm parameters: source[8]

threshold=1.5: If merging a new sample with the nearest subcluster is not less than radius 1.5 a new subcluster will launch.

branching_factor=50: In each node, the highest number of CF subclusters is 50.

n_clusters = None: The final stage of clustering is not carried out and the subclusters are kept as they are.

compute_labels=True: Calculating labels for each fit.

copy=True: Create a copy of the data issued so that the original dataset is not overwritten.

CHAMELEON algorithm

This algorithm is a combination of different algorithms.Here we are using NearestNeighbors from source[9] and AgglomerativeClustering from source[10].

NearestNeighbors parameters

n_neighbors=10: 10 nearest neibours are considered

radius=0.5: Scope space of parameters

algorithm='auto': Trying to find the most fitting algorithm

leaf_size=30: The scale of the leaf was passed on to BallTree or KDTree. The construction and query pace can be influenced by this

metric='minkowski': distance metric is minkowski

p=2: euclidean_distance for pairwise comparison

metric_params=None: Extra arguments for keywords for the metric function.

n_jobs=None: 1 Parallel task to run in the quest for neighbors

AgglomerativeClustering parameters:

n_clusters=4: 2 number of clusters for both dataset1 and dataset4 .

affinity='euclidean': linkage is "ward", so "euclidean" is set

memory=None : No caching is performed.

connectivity=None: The algorithm for hierarchical clustering is unstructured.

linkage='ward',:The variability of the clusters being combined is minimized.

distance_threshold=None: The threshold of the connection interval over which clusters are not combined

compute_full_tree='auto': Used for early termination of the construction of the tree but as distance_threshold is none so it is equivalent to false

Comparison of K means with Birch algorithm:

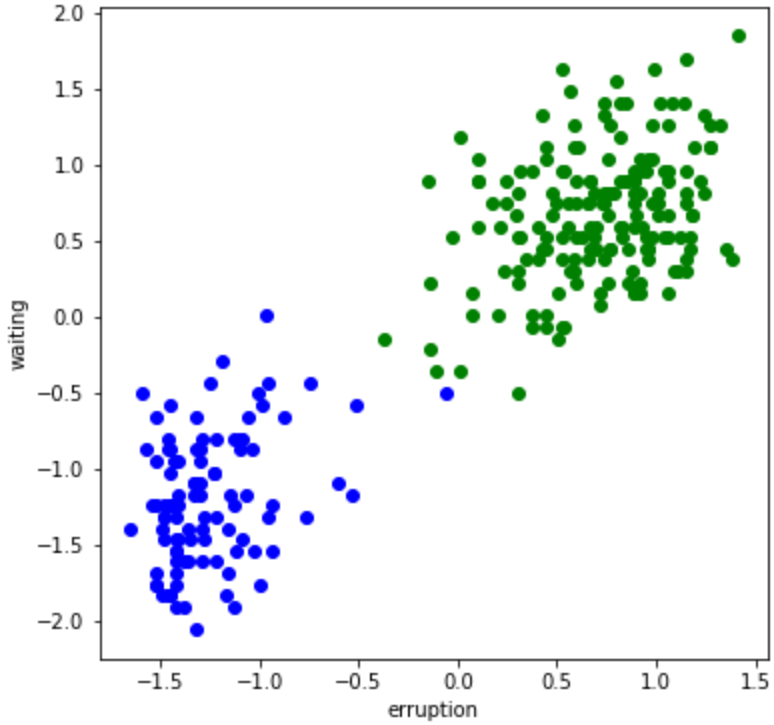


Fig 1: k means clustering algorithm on dataset1

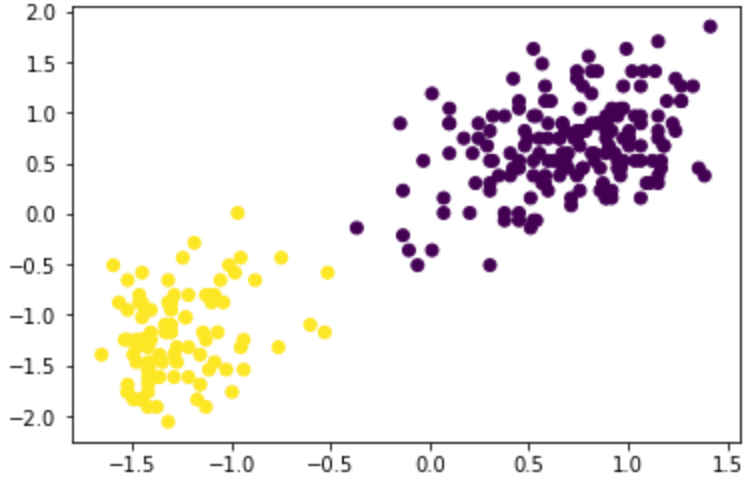


Fig 2: Birch clustering algorithm on dataset1

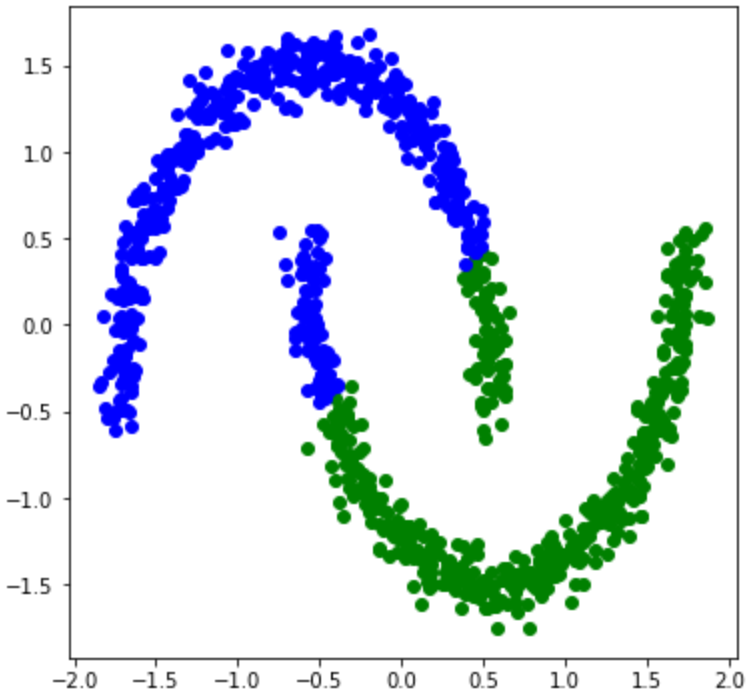


Fig 3: k means(k=2) on dataset 4

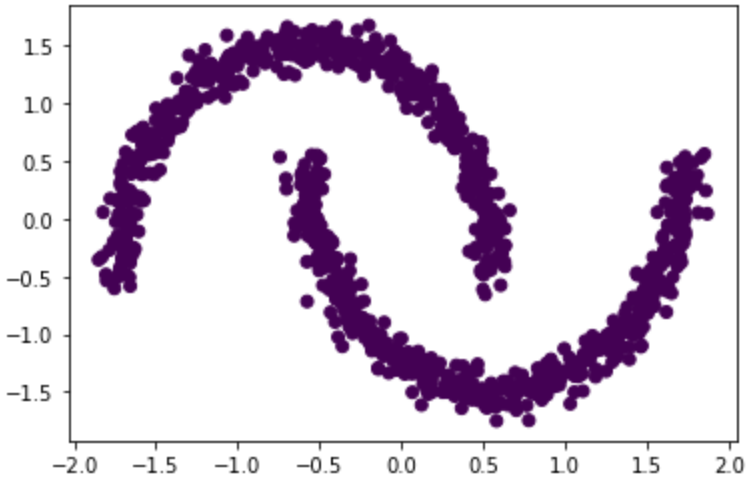


Fig: 4: Birch on dataset4

1. Tighter clusters than Birch algorithm are generated by k means clustering which can be shown using fig1 and fig2.
2. K means clustering does not perform well for different shapes and sizes as we can see in Fig1 some datapoints which should be under green clusters according to visualization,are classified as a data point in blue clusters.But these clusters are classified properly

comparatively by Birch in fig 2 .Similarly in fig3 the data points which are not inside the circle shape is not leveled properly by k means algorithm.

3. Silhouette score for Birch and k means(k=2) for dataset1 is 0.75,which tests how well a data point fits into its cluster allocated.So for dataset1 both these algorithms show similar performance.
4. ARI for k means dataset 4 with k=2 is 0.47834185309230604 and ARI for birch dataset4 is 0.0 which indicates that data points are randomly leveled.So for dataset4 k means performs better than Birch algorithm.
5. NMI for Birch dataset4 is 0.0 which means no mutual information for dataset4 when Birch algorithm is used and NMI for k means dataset4 with k=2 is 0.3802907701853083 .So for dataset4 k means performs better than Birch algorithm.

Comparison between k means and K-medoid algorithms

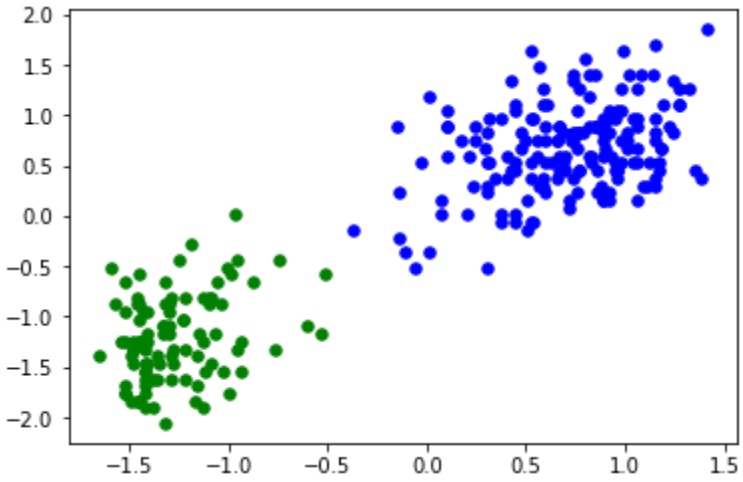


Fig:5: K-medoid(k=2) for dataset1

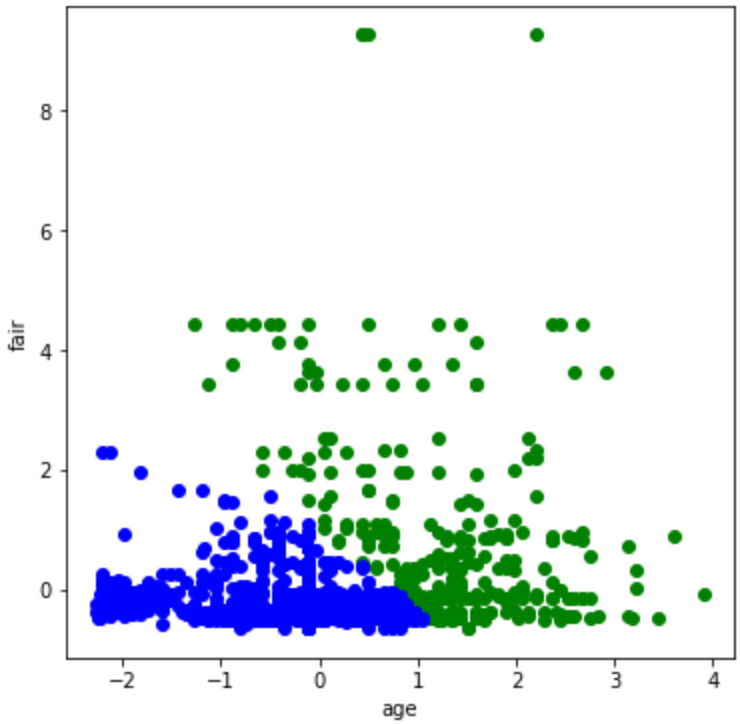


Fig 6: k means(k=2) for dataset2

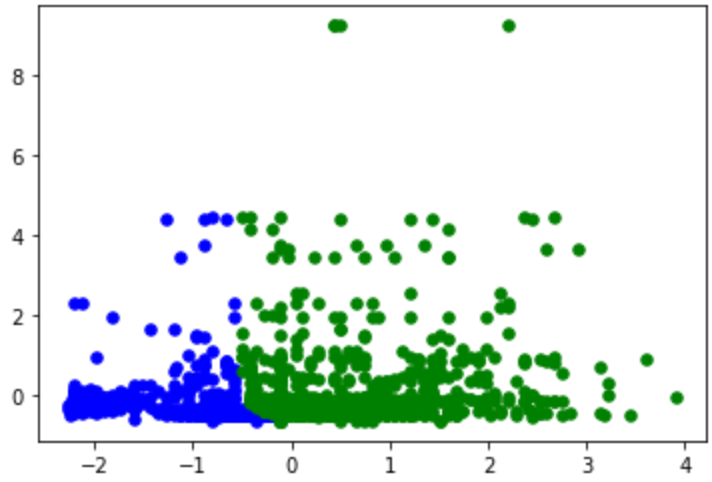


Fig 7: K-medoid(k=2) for dataset2

1. In Fig1 for k means algorithm in dataset1 the blue leveled data point which should be leveled green according to visualization is fixed in fig5 using K-medoid.The implementation idea of k medoid is generated from source[2].Here in this algorithm absolute distance is used for generating distance matrix.
2. From Fig6 it can be seen that k means does not perform well for noise and outliers but K-medoid performs well for dataset2 in Fig7 where K-medoid classified the datapoints

into two clusters comparatively as all the densely connected datapoints are in one cluster in K-medoid unlike k means for dataset2.In Fig 6 k means algorithms separate the data points into two clusters in the area where the datapoints are densely connected to each other.

3. Silhouette score for K-medoid dataset 1 with k=2 is 0.54 and for k means (k=2) it is 0.75 which shows k medoid performs less efficiently than k mean in dataset1
4. ARI for K-medoid dataset4 with k=2 is 0.4618780249161325 and ARI for k means dataset 4 with k=2 is 0.47834185309230604 which are similar ,so both of these algorithms perform similarly for dataset4
5. NMI for K-medoid dataset4 with k=2 is 0.3856484227821514 and NMI for k means dataset4 with k=2 is 0.3802907701853083 which are similar ,so both of these algorithms have similar mutual information and they show similar performance for dataset4

Comparison of AGNES with other clustering algorithms

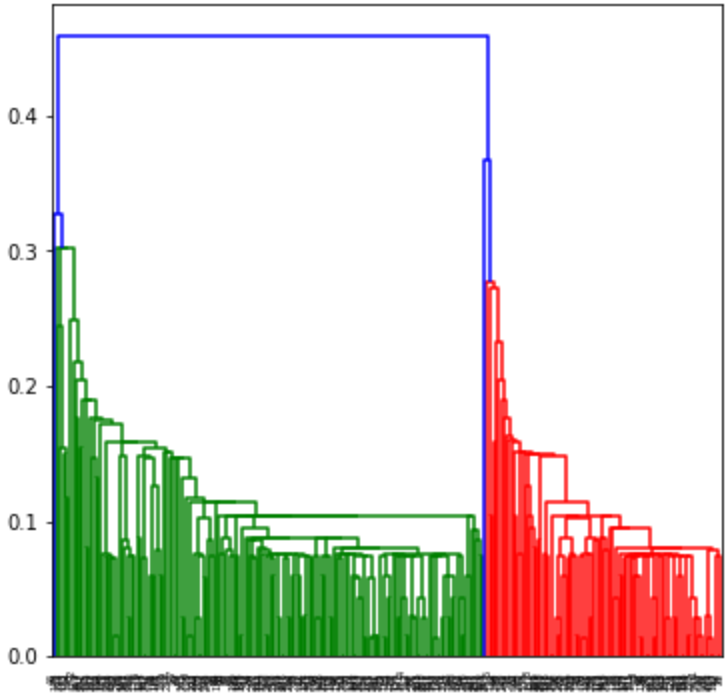


Fig 8: AGNES algorithm for dataset1

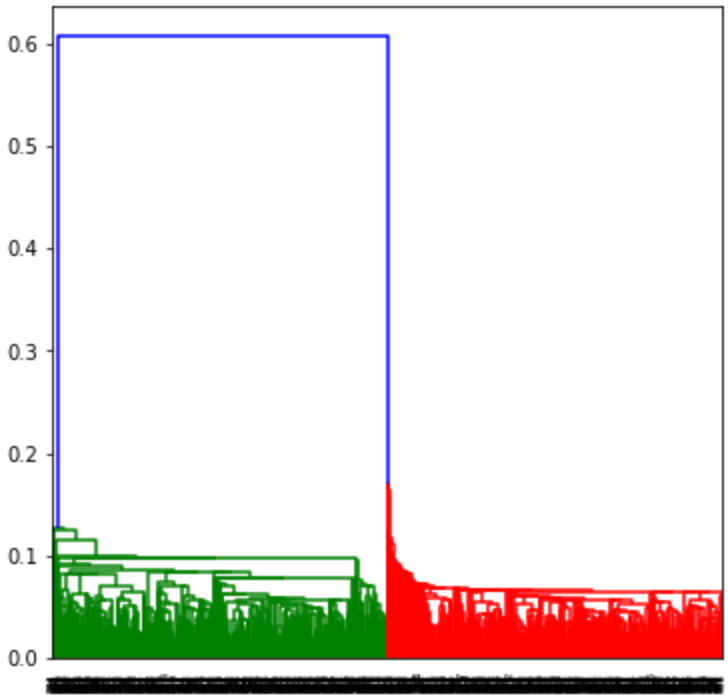


Fig:9: AGNES algorithm for dataset4

1. The AGNES algorithm in Fig:8 starts with all 273 datapoints for dataset1 in a cluster and divides them until the number of samples become 262.The implementation idea of AGNES is generated from source[3] and [4].Here in this algorithm euclidean distance is used for generating distance matrix .Here also the datapoints are divided into two clusters but unlike k means algorithm number of clusters need not to be defined.For dataset4 in Fig 9 the AGNES algorithm starts with 1000 datapoints and divides the data points in clusters until number of data points become 996.
2. From fig8 it can be interpreted that a comparatively larger number of datapoints are in green cluster and less number of datapoints are in red cluster.So the proportion of datapoints in the clusters have similarity with the original distribution of data points in dataset1 and for dataset4 almost similar amount of data points are in both clusters which also reflects the original proportion of data points in the original data points in dataset4.
3. The number of clusters need not be defined in AGNES, unlike k means algorithm and K-medoid algorithm.
4. Birch algorithm performs better for larger datasets compared to AGNES
5. AGNES algorithm does not have any sensitive parameters other than terminating conditions as DBSCAN algorithm.
6. AGNES algorithm performs similar CHAMELEON for dataset4 as it can successfully partition the dataset into two different clusters with an exception of a few data points which are in the middle of two clusters.

Comparison between k means and DBSCAN algorithm

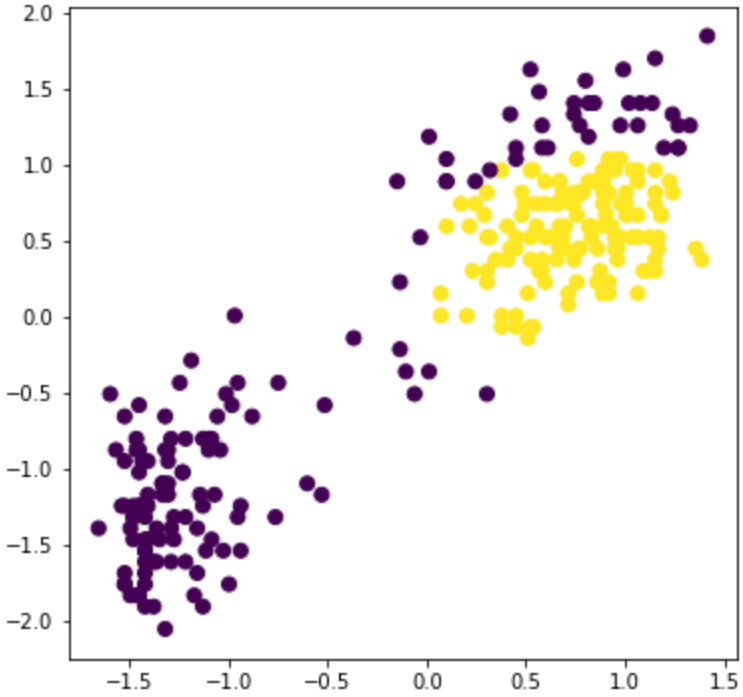


Fig10 : DBSCAN algorithm with minpoint 3 and eps=0.9 on dataset1

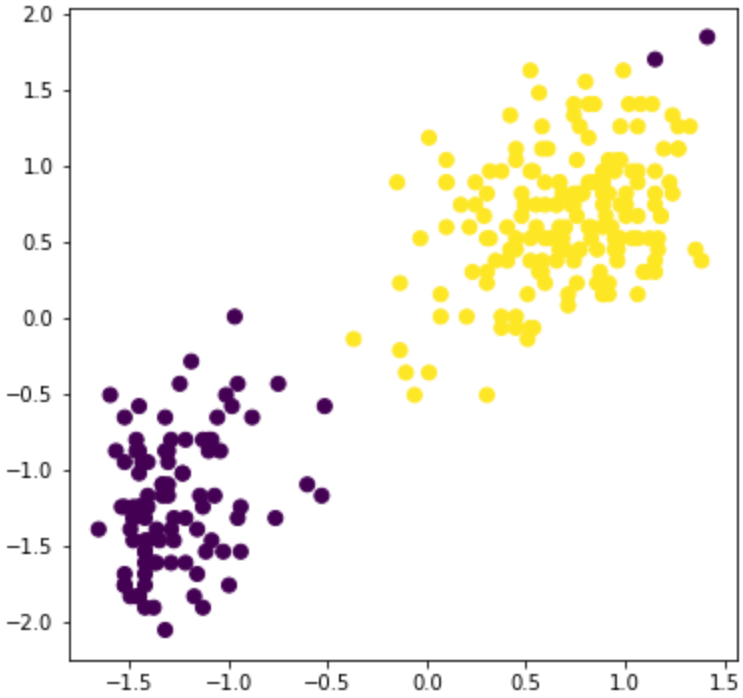


Fig11 : DBSCAN algorithm with minpoint 3 and eps=1.5 on dataset1

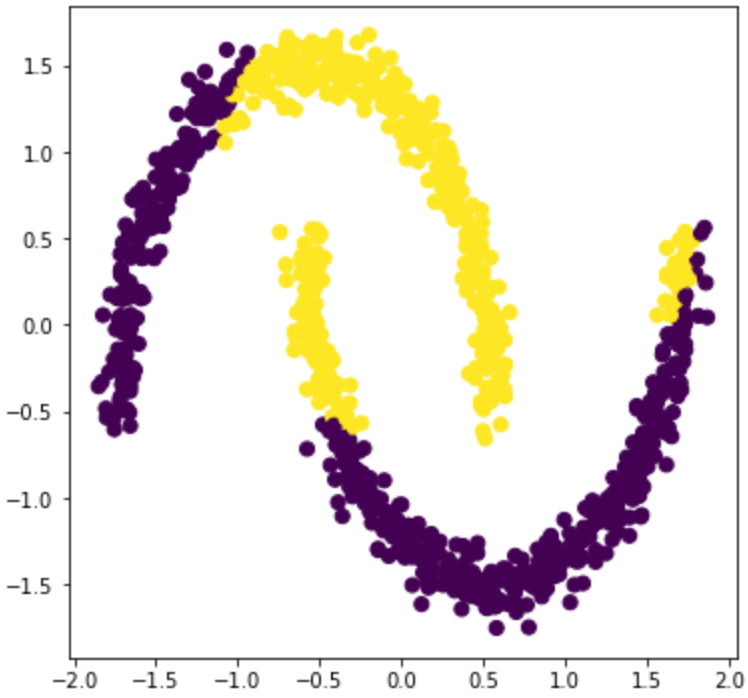


Fig 12: DBSCAN algorithm with minpoint 3 and eps=1.5 on dataset4

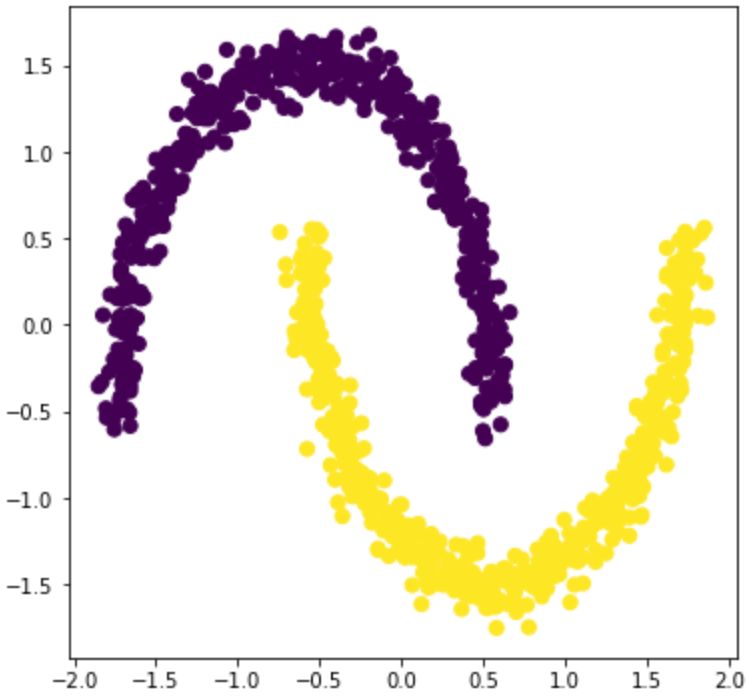


Fig 13: original dataset4

1. The implementation idea of DBSCAN is generated from source [5] and [6].The main challenge in DBSCAN algorithm is to find the appropriate minimum point and eps for any dataset.For dataset 1 in fig10 when the minimum point is 3 and eps=0.9 then all the data points are not leveled correctly according to visualization but when eps = 1.5 with the same minimum point fig 11, it correctly classified most of the points. Similarly for dataset4 the original dataset4 looks like fig13 but using the DBSCAN algorithm on dataset4 with minimum points = 3 and eps=1.5 it could not level all the data points correctly in fig 12.
2. It need not specify the number of clusters, unlike k means clustering.
3. Silhouette score for DBSCAN dataset 1 is 0.73 and for k means it is 0.75,so k means performs better than DBSCAN for dataset1.
4. ARI for DBSCAN dataset4 is 0.15126659647188057 and ARI for k means dataset 4 with k=2 is 0.47834185309230604 so k means performs better than DBSCAN for dataset4
5. NMI for DBSCAN dataset4 is 0.3802907701853083 and NMI for k means dataset4 with k=2 is 0.3802907701853083 which indicates that both of these algorithms have similar mutual information.
6. DBSCAN has successfully detected the noise for dataset 1 unlike k means in fig11.

Comparison between K-medoid and Birch

1. Silhouette score for K-medoid dataset 1 with k=2 is 0.54 and for Birch it is 0.75,so Birch performs better than K-medoid for dataset1
2. ARI for K-medoid dataset4 with k=2 is 0.4618780249161325 and it is 0 for Birch so K-medoid perform better than Birch algorithm for dataset4
3. NMI for K-medoid dataset4 with k=2 is 0.3856484227821514 and 0 for Birch so for dataset4 K-medoid performs better than Birch algorithm
4. From Fig2 and fig 5 we can see that both of these algorithms Birch and k medoid show almost similar results in the scatter plot.

Comparison between K-medoid and DBSCAN

1. Silhouette score for DBSCAN dataset 1 is 0.73 and Silhouette score for K-medoid dataset 1 with k=2 is 0.54, so DBSCAN performs better than K-medoid for dataset1

2. ARI for DBSCAN dataset4 is 0.15126659647188057 and ARI for K-medoid dataset4 with k=2 is 0.4618780249161325 so K-medoid perform better than DBSCAN algorithm for dataset4.
3. NMI for DBSCAN dataset4 is 0.3802907701853083 and NMI for K-medoid dataset4 with k=2 is 0.3856484227821514 which indicates that both of these algorithms have similar mutual information.
4. For K-medoid algorithm the number of cluster size need to be determined but it has no sensible parameters as DBSCAN.

Comparison between Birch and DBSCAN

1. Silhouette score for Birch dataset 1 is 0.75 and silhouette score for DBSCAN dataset 1 is 0.73 so Birch performs better than DBSCAN for dataset1
2. ARI for DBSCAN dataset4 is 0.15126659647188057 but for Birch it is 0 so DBSCAN performs better than Birch for dataset4
3. NMI for DBSCAN dataset4 is 0.3802907701853083 but for Birch it is 0 so DBSCAN performs better than Birch for dataset4
4. DBSCAN could identify the noise from dataset 1 .

Comparison of k means with CHAMELEON algorithm

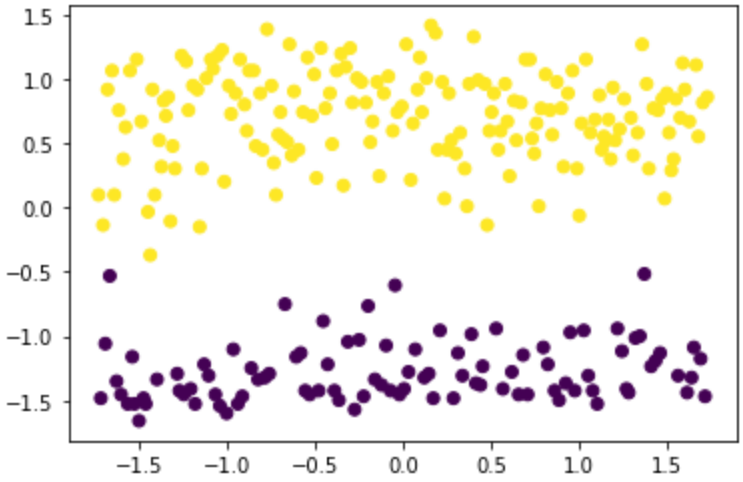


Fig :14: CHAMELEON on dataset1

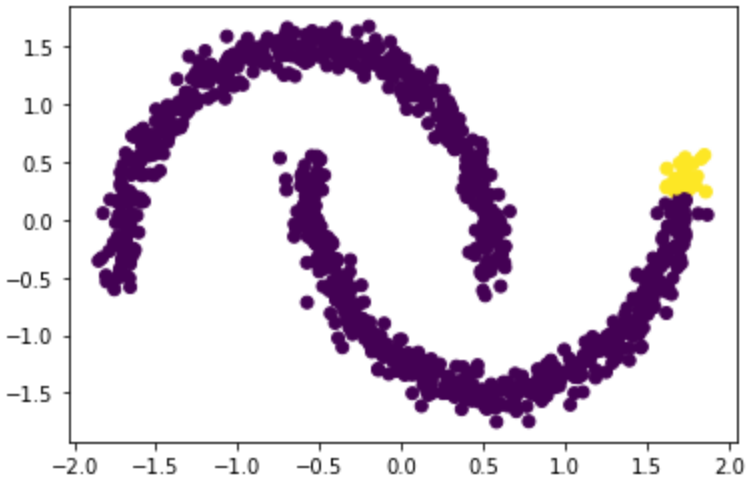


Fig: 15: CHAMELEON on dataset4

1. In fig 14 the CHAMELEON algorithm is able to classify dataset1 into two correct clusters properly unlike (figure 1) k means algorithm.As some points from upper clusters are classified incorectly as datapoints of lower cluster in fig14 similar to k means in fig1.
2. In figure 15 CHAMELEON did not perform well than K means algorithm in fig3.
3. ARI for CHAMELEON dataset4 is 0.0026028665264942403 and ARI for k means dataset4 with k=2 is 0.47834185309230604 so k means performed better for dataset4
4. NMI for CHAMELEON dataset4 is 0.04514919641021516 and NMI for k means dataset4 with k=2 is 0.3802907701853083 so k means performed better for dataset4

Comparison of K-medoid with CHAMELEON algorithm

1. In fig 14 the CHAMELEON algorithm is unable to classify dataset1 into two correct clusters in comparison with (figure 5) k medoid algorithm performed better than CHAMELEON as it successfully classified the dataset into two levels.
2. ARI for KMediod dataset4 with k=2 is 0.4618780249161325 and ARI for CHAMELEON dataset4 is 0.0026028665264942403 so k mediod performed better than CHAMELEON for dataset4
3. NMI for KMediod dataset4 with k=2 is 0.3856484227821514 and NMI for CHAMELEON dataset4 is 0.04514919641021516 so k mediod performed better than CHAMELEON for dataset4

Comparison of Birch with CHAMELEON algorithm

1. In fig 14 the CHAMELEON algorithm is unable to classify dataset1 into two correct clusters properly compared to Birch. In figure 2 Birch algorithm performed better than CHAMELEON as it successfully classified the dataset into two levels.

2. ARI for birch dataset4 is 0.0 and ARI for CHAMELEON dataset4 is 0.0026028665264942403 so for dataset4 CHAMELEON performed better than Birch,which is also visible from fig 4 and fig 15.As CHAMELEON partially classify one cluster .
3. NMI for Birch dataset4 is 0.0 and NMI for CHAMELEON dataset4 is 0.04514919641021516 so for dataset4 CHAMELEON performed better than Birch.

Comparison of DBSCAN with CHAMELEON algorithm

1. Silhouette score for DBSCAN dataset 4 is 0.22 and silhouette score for CHAMELEON dataset 4 is 0.14 so DBSCAN performed well for dataset4 than CHAMELEON which is visible from fig 12 and fig 15 as in fig 12 a larger part of yellow cluster is present than for CHAMELEON in fig 15.
2. ARI for DBSCAN dataset4 is 0.15126659647188057 and ARI for CHAMELEON dataset4 is 0.0026028665264942403 so DBSCAN performed better than CHAMELEON in dataset4.
3. NMI for DBSCAN dataset4 is 0.3802907701853083 and NMI for CHAMELEON dataset4 is 0.04514919641021516 so DBSCAN performed better than CHAMELEON in dataset4.
4. With minpoint 3 and eps=1.5 DBSCAN performed better than CHAMELEON as it detected outliers from dataset1 in fig 11 but in fig 14 some datapoints are incorrectly classified according to human visualization.So DBSCAN performed better than CHEMELEON for dataset1. CHEMELEON could not detect outliers properly which points are in middle of two clusters.

So we can conclude that in this assignment we have performed a comparative analysis among six clustering algorithms.

References:

1. https://realpython.com/k-means-clustering-python/
2. https://github.com/shenxudeu/K_Medoids
3. https://medium.com/@darkprogrammerpb/agglomerative-hierarchial-clustering-from-scratch-ec50e14c3826
4. https://github.com/Darkprogrammerpb/DeepLearningProjects/tree/master/Project40/agglomerative_hierarchial_clustering
5. https://medium.com/@darkprogrammerpb/dbscan-clustering-from-scratch-199c0d8e8da1
6. https://github.com/Darkprogrammerpb/DeepLearningProjects/tree/master/Project41/DBSCAN
7. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html
10. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html