Programming Assignment 3

Sankeerth.D
EE13B102
Electrical Engineering

November 16, 2016

Abstract

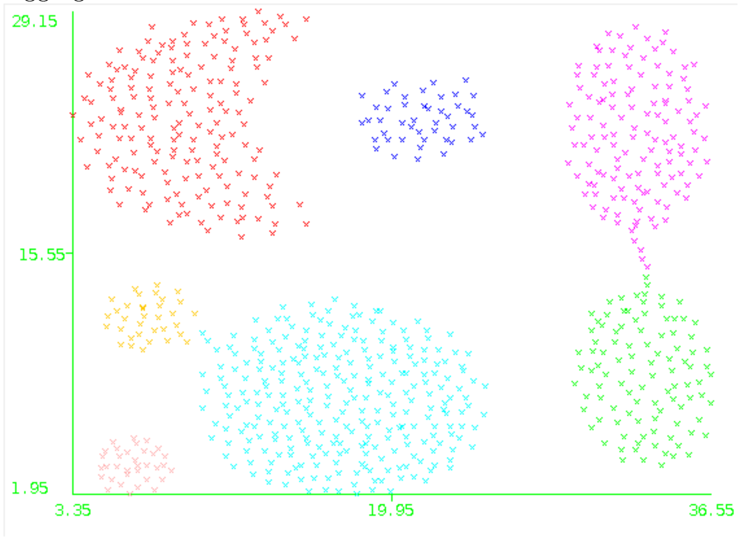
For different configurations of datapoints in 2D space, we'll be running K-means, DBSCAN and heirarchical clustering algorithms and try to justify their behavior on each dataset.

1 Convert to ARFF:

The data is converted into arff format using the weka GUI converter provided along with weka.

2 Visualizing the datasets:

Weka's visualizer tab is used to display the datapoints in 2D plane:
Aggregation:



Kmeans: Kmeans will not be able to recover all points into specific clusters, as the sizes of the clusters have significant differences, but would give a cluster assignment that will make sense. The red and the blue clusters will merge to an extent.

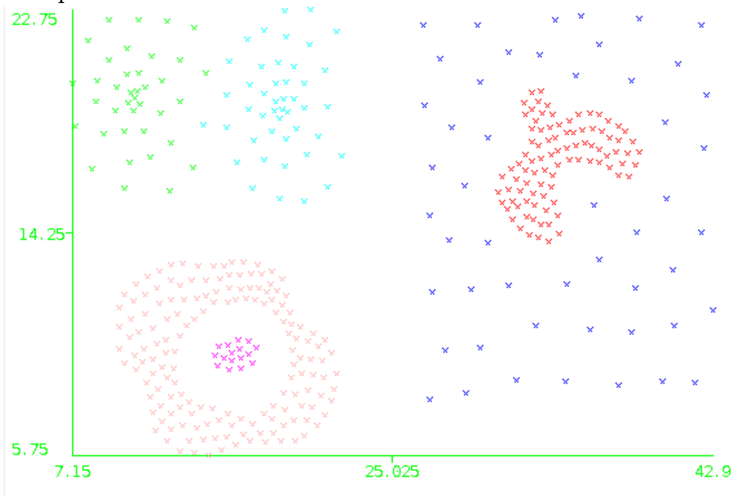
DBSCAN: DBSCAN works well in this case, on choosing a small enough epsilon and atleast 4-5 minPoints, as the density of each cluster is fairly constant and clusters are separated. On choosing 1-2 minpoints, the narrow path would be connected and clusters will be merged.

Single-Link Heirarchical clustering: Due to the points being connected by narrow paths, Single link cluster will merge the purple ang green points into a single cluster in the very base of the dendrogram. Hence performance will not be as good as it merges the clusters connected by the narrow path.

Complete-Link Heirarchical Clustering:This would work fine in the above case and return the correct cluster assignments.

'

Compound:



Kmeans: KMEANS cannot detect clusters which have a hollow nature, i.e., one inside the other. Kmeans returns a random output with this dataset.

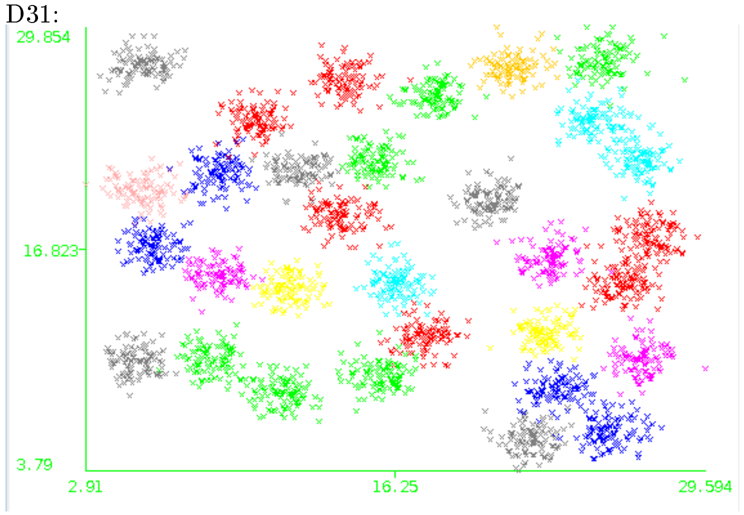
DBSCAN: DBSCAN faces an issue with clustering this dataset. The purple and pink clusters can be recovered, but The other clusters, blue and red, cannot be recovered. This is due to the fact that DBSCAN forces a threshold on density. While it is clustering the blue class, the red cluster will surely merge to be a part of it. Same goes for the light-blue and green clusters, as they have the same density and mix up well.

Single-Link Heirarchical clustering: Single link clustering Correctly clusters the pink and purple clusters, but would not be able to recover the blue cluster because as we go up the dendrogram, blue is merged with the red.

Complete-Link Heirarchical Clustering: This will not be able to get the correct cluster assignment, even for the pink and purple datapoints. Also, it cannot get correct output for the red-blue clusters, as the classes aren't tight.

’

D31:



Kmeans: This will recover the clusters partially for higher k, but will split the acual classes. For lower k, the classes whih are close are merged and outliers are assigned a separate class. Hence clusters can be partially recovered.

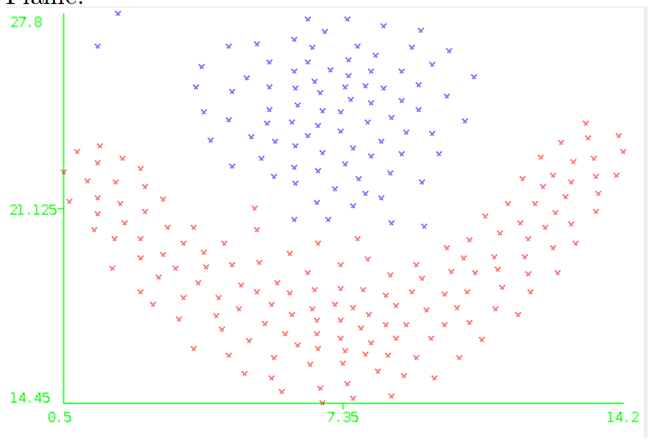
DBSCAN: It performs reasonably well in recovering most of the clusters, but fails to do those which overlap to an extent, as DBSCAN can’t see any change in density in that case.

Single-Link Heirarchical clustering: The clusters are close together and single link might merge two seperate clusters together in early stages itself. Hence it doesn’t perform well with this dataset.

Complete-Link Heirarchical Clustering: This is able to recover a major portion of the clustering and worked better than kmeans and DBSCAN. All the clusters are tight and spherical, and almost all clusters are returned on running this.

’

Flame:



Kmeans: KMeans will mix up both the clusters as the red-cluster because of the nonspherical shapes of each of the clusters.

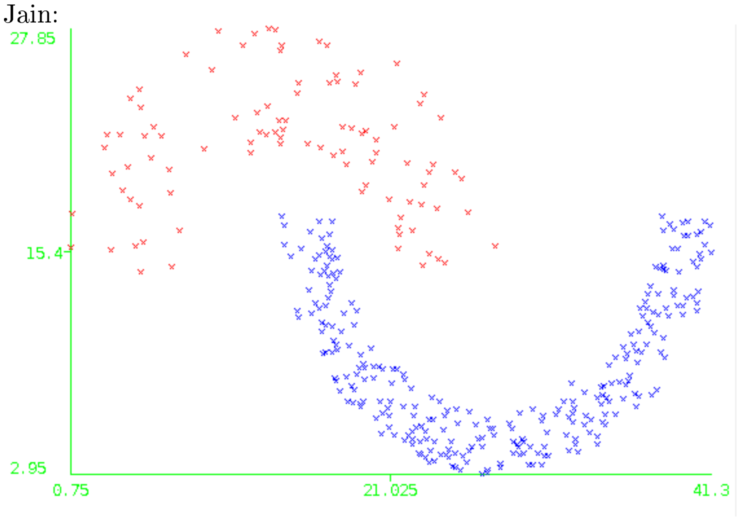
DBSCAN: The density of both the clusters is the same and they mix in well. DBSCAN won't be able to distinguish between the two because it can't see any change in density of the regions.

Single-Link Heirarchical clustering: Will not work at all if we initially assign number of clusters = 2, due to the high noise in data, and mixing. At num_clusters=13, we will recover most parts of the clusters, with extra outlying clusters by the side.

Complete-Link Heirarchical Clustering: This doesn't work well as the clusters returned by this are tight, unlike in the dataset which has long and curved clusters. We would get a low accuracy on running complete link clustering.

'

Jain:



Kmeans:Due to the irregular shape, k-means will be unable to return reasonable result in this case.

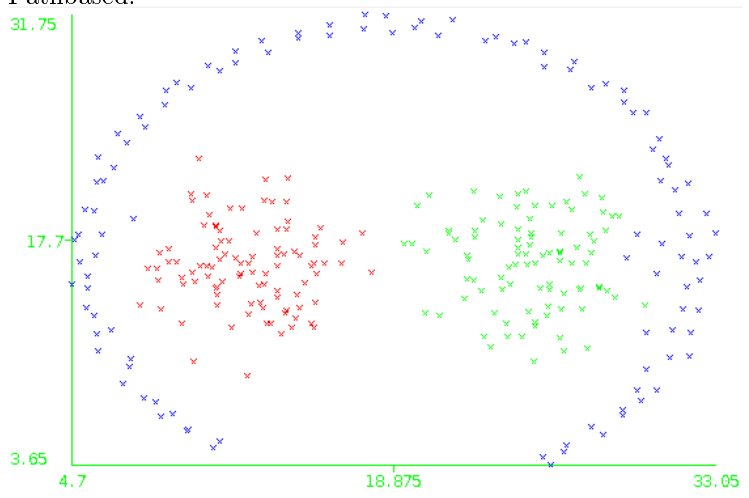
DBSCAN:DBSCAN will be able to separate out the clusters, as done in the third section. Although the density isn't the same, the class separation is high enough to not merge the clusters together.

Single-Link Heirarchical clustering: It does not work at all if we as k for 2 clusters due to the noise in data. On setting the number of clustersto 12, we can recover most part of the clusters, with a few extra singleton clusters.

Complete-Link Heirarchical Clustering: As the clusters atr long and not tight, this doesn't output a reasonable split. The clusters returned do not represent the classes

'

Pathbased:



Kmeans:Kmeans gives unreasonable results as irregular shapes are not captured by the kmeans algorithm

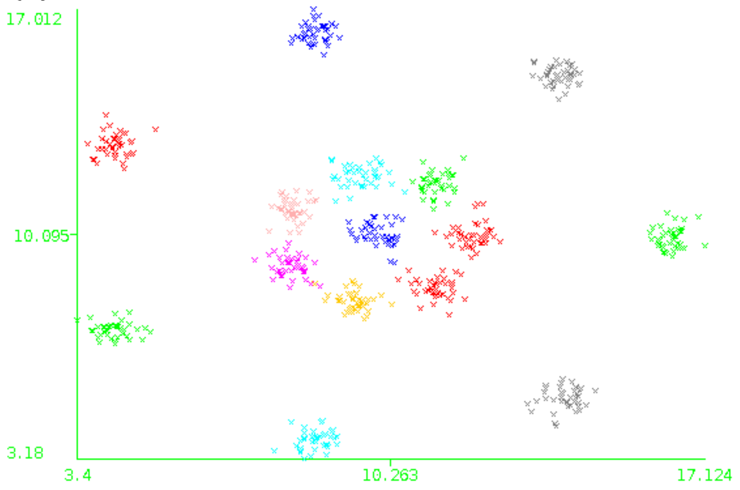
DBSCAN: The green and blue cluster points are very close to each other and DBSCAN can't separate out the two. The red cluster will be separated from the others, and the green and blue clusters are only partially recovered.

Single-Link Heirarchical clustering: The data has many noisy elements and the distance making the minimum distance between each cluster small for single link. It will not work well in this case if we set the number of clusters to 3. As the distances between clusters and between points in clusters is same, single link won't give good results.

Complete-Link Heirarchical Clustering: It will merge the clusters in a wrong way, because the distance between clusters and between points in clusters is comparable.

'

R15:



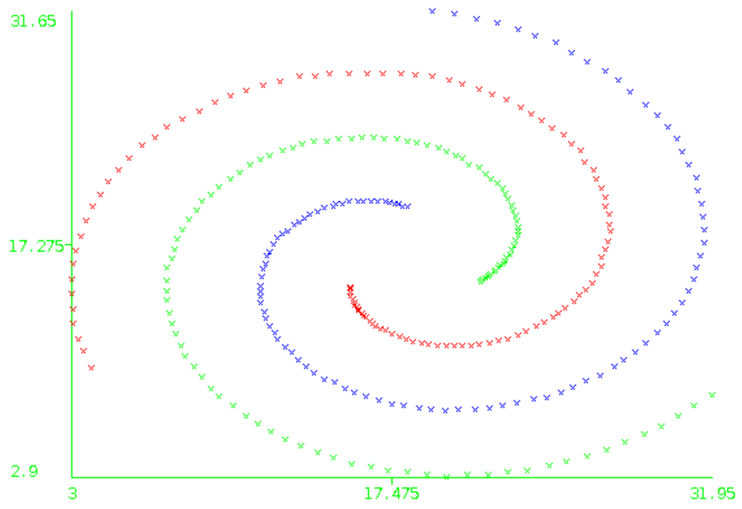
Kmeans: With the correct value of k given, k means performs well with this dataset as the classes appear to be close to a gaussian mixture.

DBSCAN:DBSSCAN will work well with the given dataset, as all the clusters have a higher and equal density in their region and there is not much overlap. Appropriate choice of epsilon and minPts will recover the clusters.

Single-Link Heirarchical clustering: This tends to merge together clusters at the center and instead assign outliers as different clusters.

Complete-Link Heirarchical Clustering: This method works very well in this case and all the clusters are recovered. All the cluters are tight and close enough for complete links to form near the base of the dendrogram.

'



Kmeans: Kmeans cannot recognize clusters of this form at all, and performs very poorly. Only gaussian-type distance based clusters are recognized by Kmeans. On running K-means on this data, we get abstract split, fat from the actual spiralling clusters.

DBSCAN: DBSCAN performs reasonably well for this dataset. On setting min-points 1 or 2 (lower value), we would be able to recover all three clusters for a range of epsilon distances.

Single-Link Heirarchical clustering:single link clustering works reasonably well. On biulding the dendrogram, the height drastically increases before merging the clusters together, and all the points which are almost in line get connected mear the very base of the dendrogram

Complete-Link Heirarchical Clustering: Complete link heirarchical clustering would face problems in outputting the correct clusters. On going up the dendrogram, the clusters form correctly, then start to get assigned to the wrong spiral.

3 K-Means on R15 dataset:

The file characterize_purity.py in the directory plots the accuracy versus number of clusters and purity versus number of clusters. Run command:

python characterize_purity.py

The KMeans_R15 directory contains the output reported on running k from 1 to 20. Also, the k=8 case has its model file in the directory.

k=8 case:

Accuracy: 53.33%

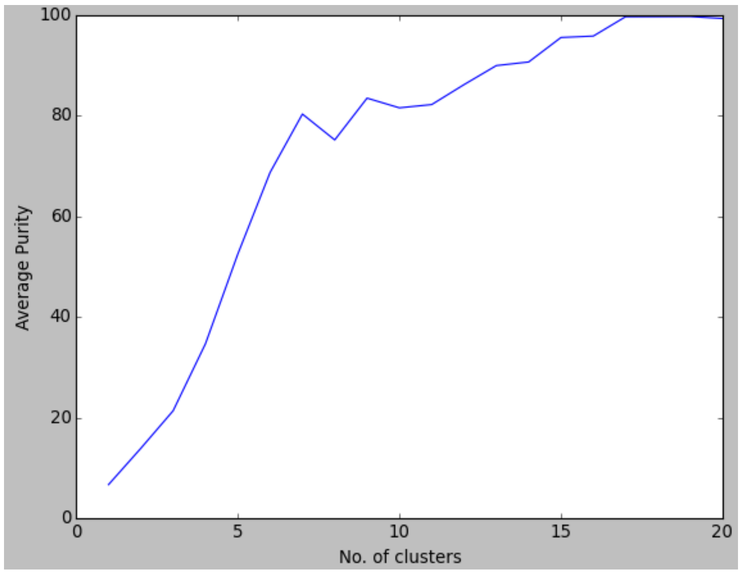
Purity of the eight clusters:

1	0.21	1	1	1	1	0.5	0.3

Average purity:75.21%

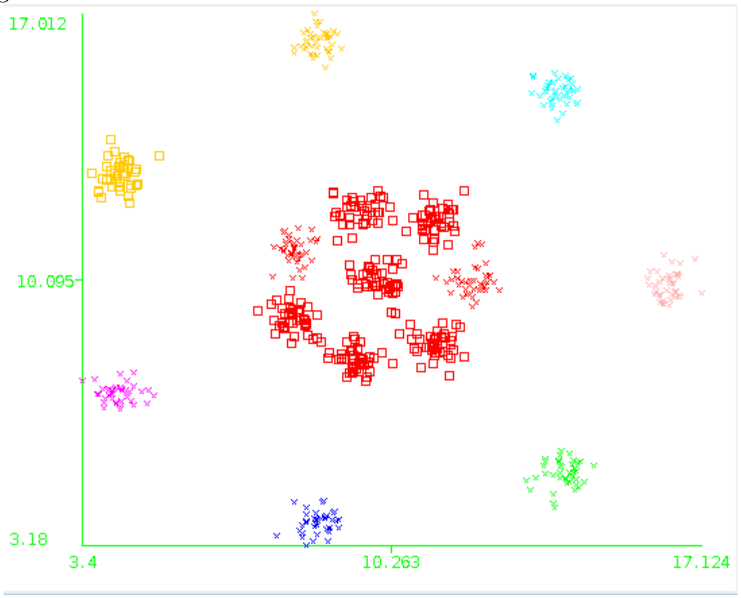
’

On increasing k from 1 to 20, the average purity of the clusters varies as shown below:



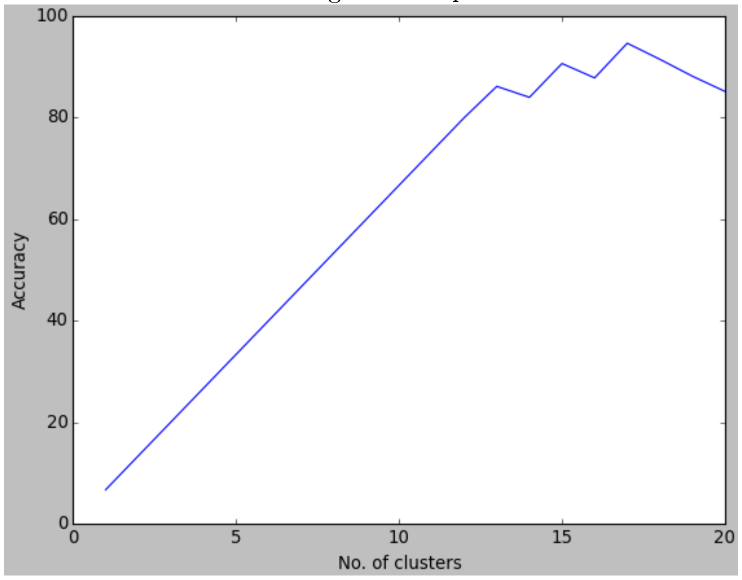
,

As we can see, on increasing k, the purity increases. This is because as k increases, we will have fewer points in each cluster, and they will be small enough to accommodate all points of the same class without extending to neigboring classes. Hence average purity increases. At k=8, however, we observe a sharp fall in the average purity because The entire center portion is merged into a single cluster:



,

The accuracy, however, on increasing k from 1 to twenty peaks at 15 which is the actual number of classes given as input.



4 DBSCAN on JAIN dataset:

The file characterize_purity.py plots the accuracy and average purity versus epsilon. Run command:

python characterisze_purity.py <m>

Where m = 1 or 3 or 6 or 9

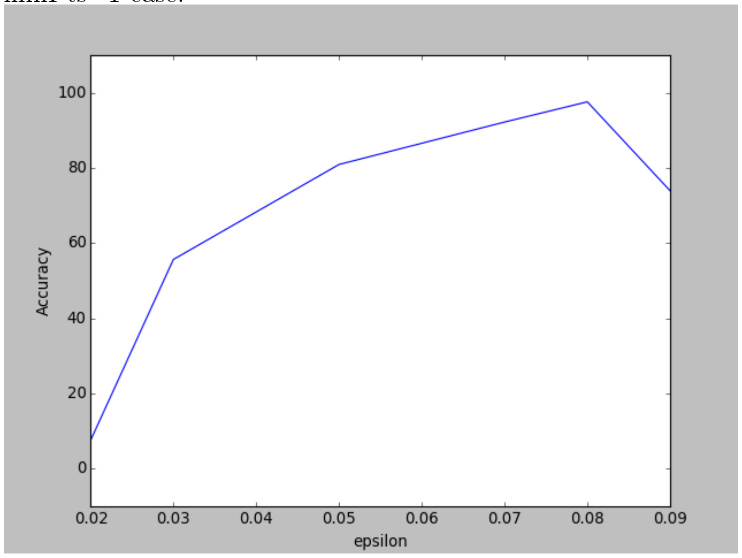
On running DBSCAN on the jain dataset, the best performance is obtained on selecting m = 3 and e = 0.08, as can be seen from the plots below. Also, just by visualizing the clustering at m=3, e = 0.08, we get 3 clusters, 2 of which mostly seoparate both the classes.

With these parameters, cluster purity is nearly 100%, and accuracy is at 90%.

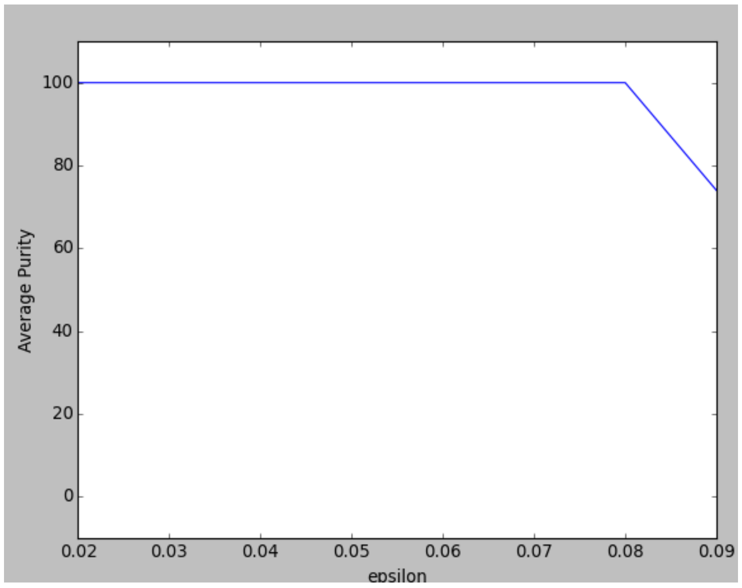
ACCURACY: Note that for lower values of epsilon, many points are still unclassifed. These unclassified points are taken to be falsely classified while calculating accuracy.

The purity observed and the accuracy of clustering vary with epsilon as shown below:

minPts=1 case:

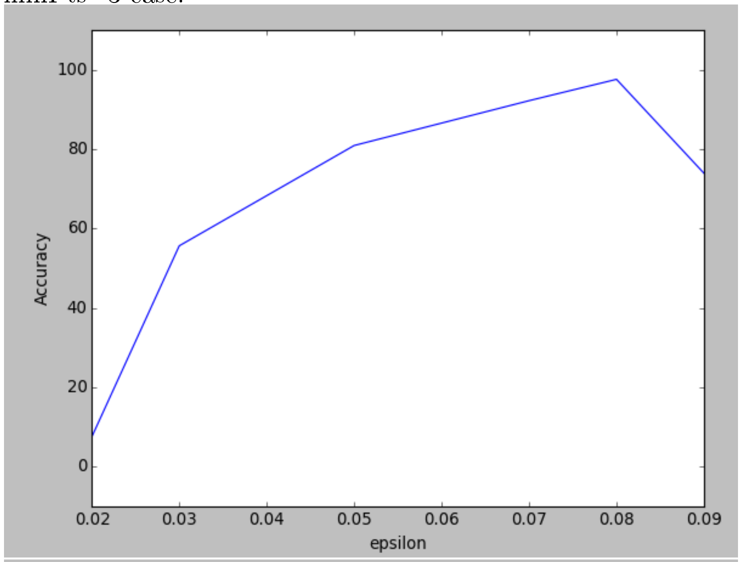


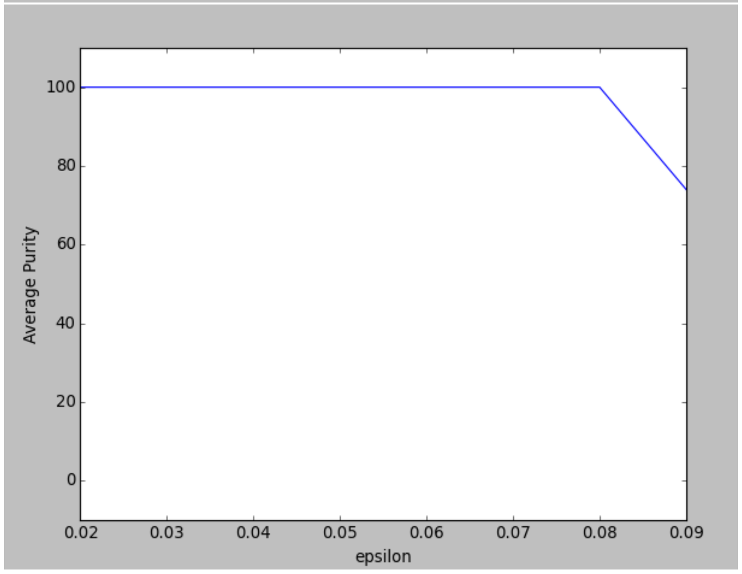
,



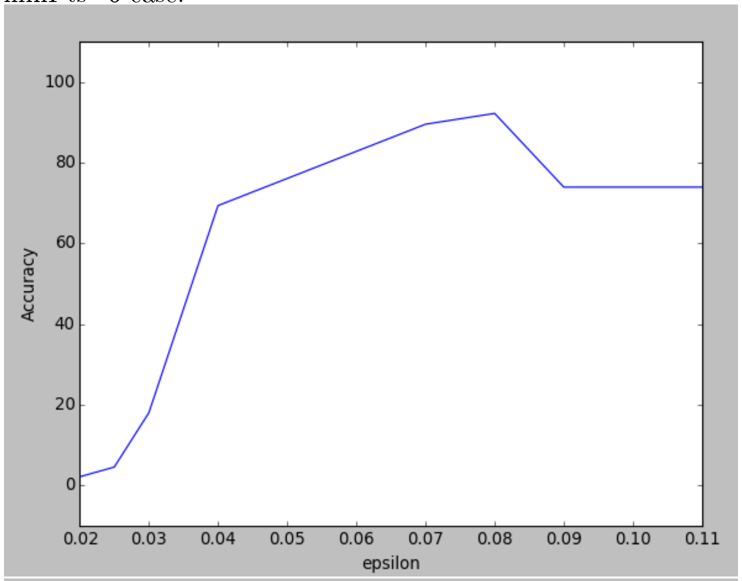
,

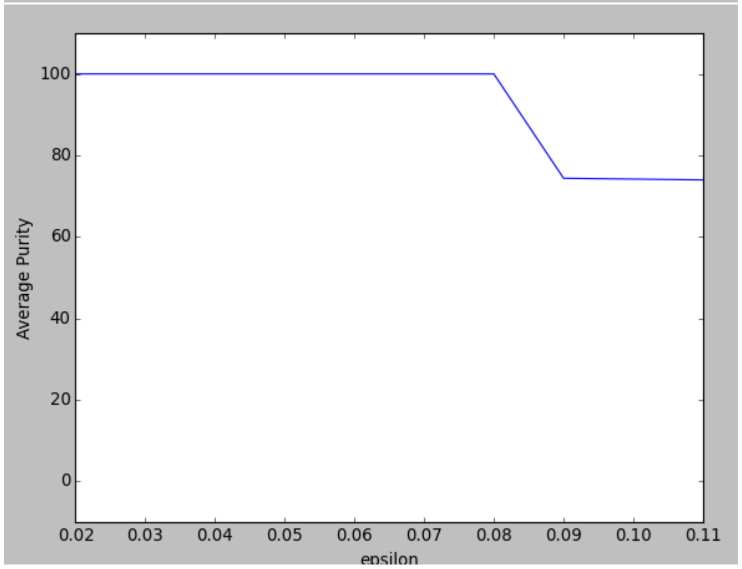
minPts=3 case:



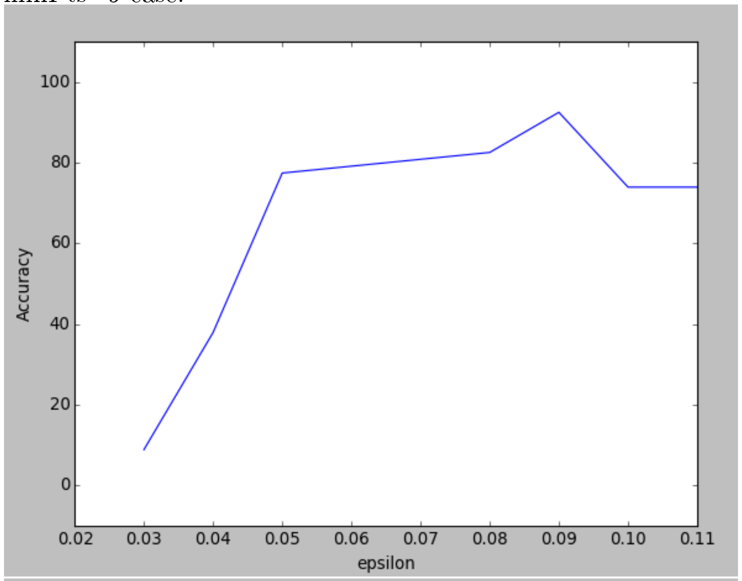


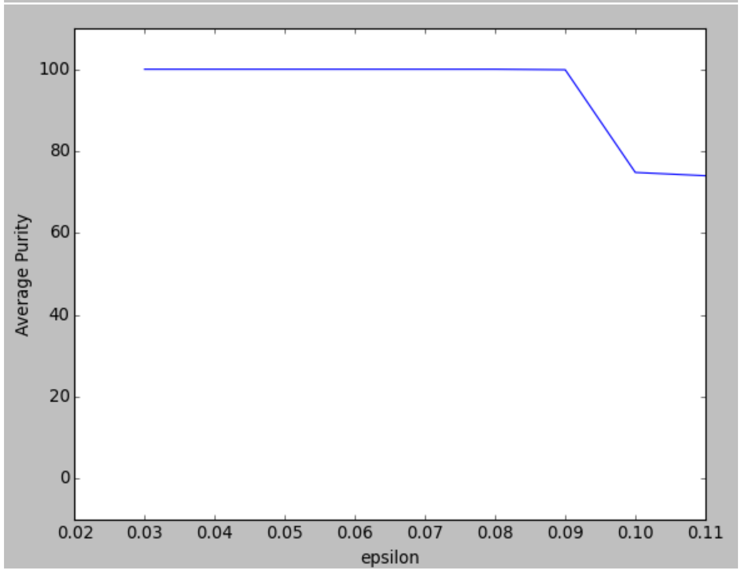
minPts=6 case:





minPts=9 case:





,

5 DBSCAN and Heirarchical clustering on path-based, spiral and flame:

The best performance on varying minpoints and epsilon

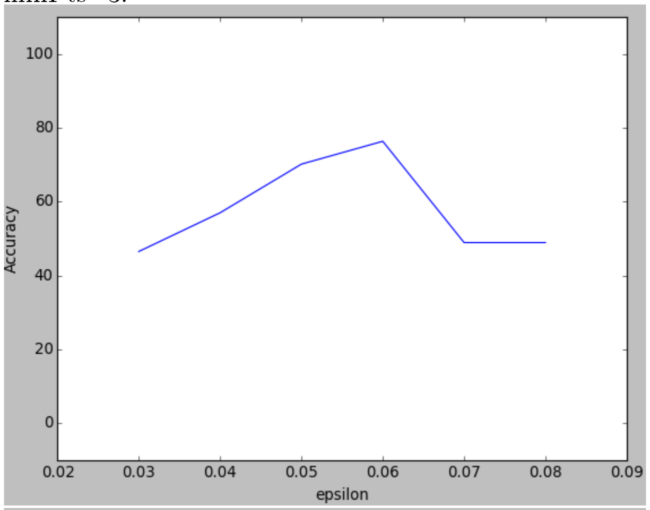
PATHBASED:

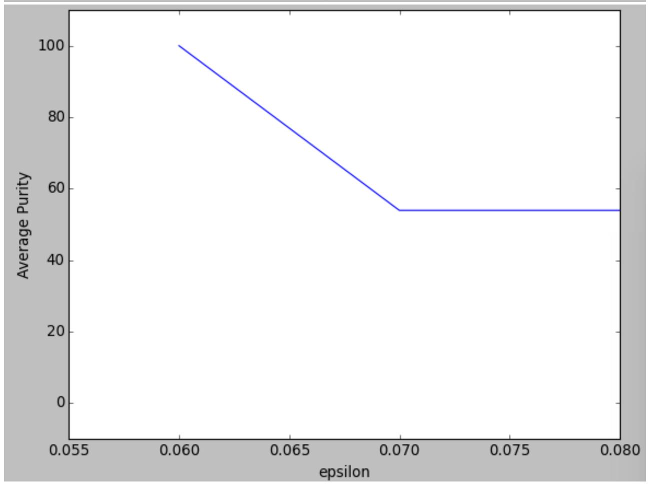
DBSCAN: Running DBSCAN on path based gave its best performance at

On running DBSCAN at minpoints 1, 3, 6, 9, the following characteristics with threspect to Eps are observed. The best accuracy is achieved at minPts = 3, Eps=0.06

Best accuracy = 78%

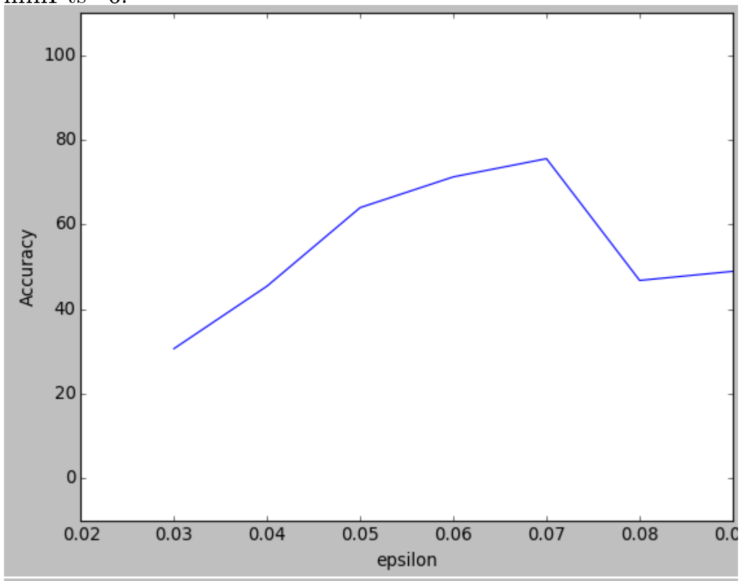
minPts=3:



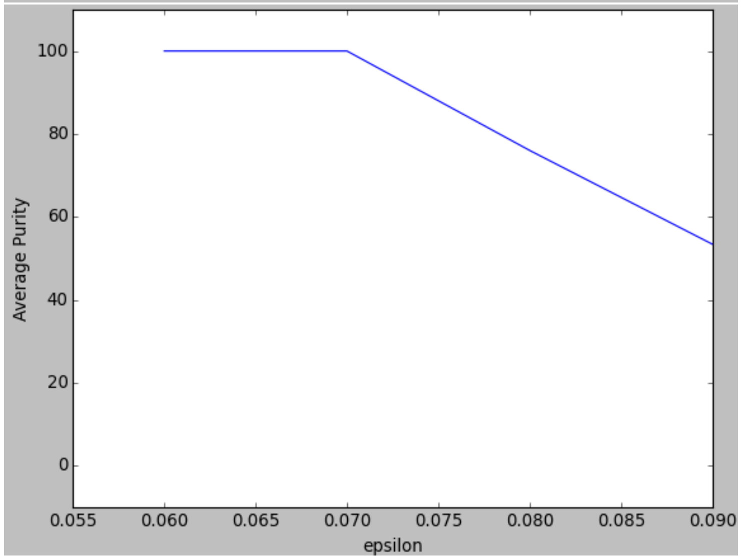


,

minPts=6:

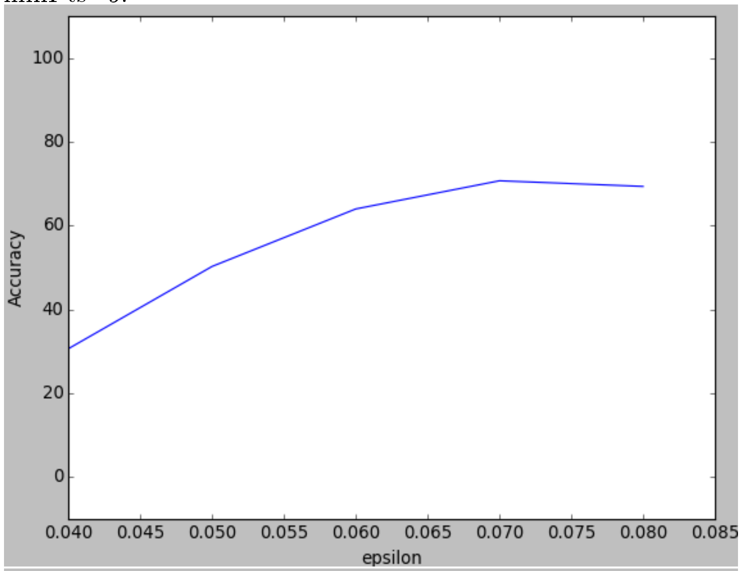


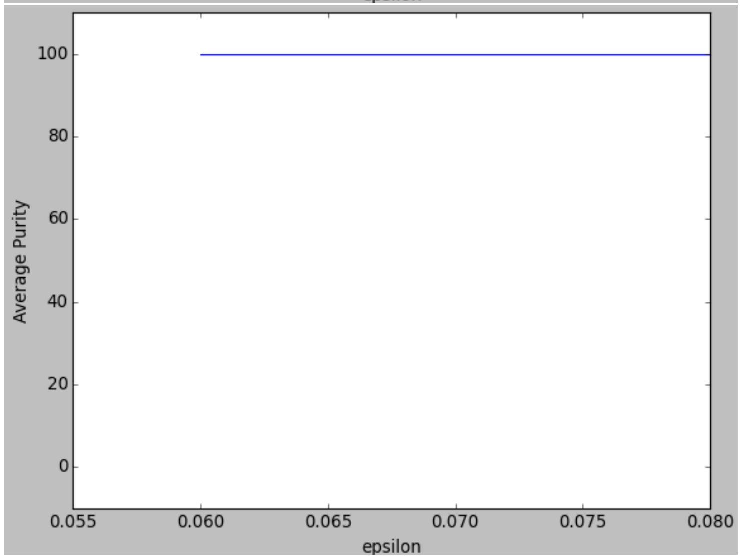
Accuracy
100
80
60
40
20
0
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.0
epsilon



Average Purity
100
80
60
40
20
0
0.055
0.060
0.065
0.070
0.075
0.080
0.085
0.090
epsilon

minPts=9:


Accuracy
epsilon
100
80
60
40
20
0
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0.075
0.080
0.085


Average Purity
epsilon
100
80
60
40
20
0
0.055
0.060
0.065
0.070
0.075
0.080

,

Heirarchical Clustering: The average and WARD distance functions for heirarchical clustering give close to the best clusteering.

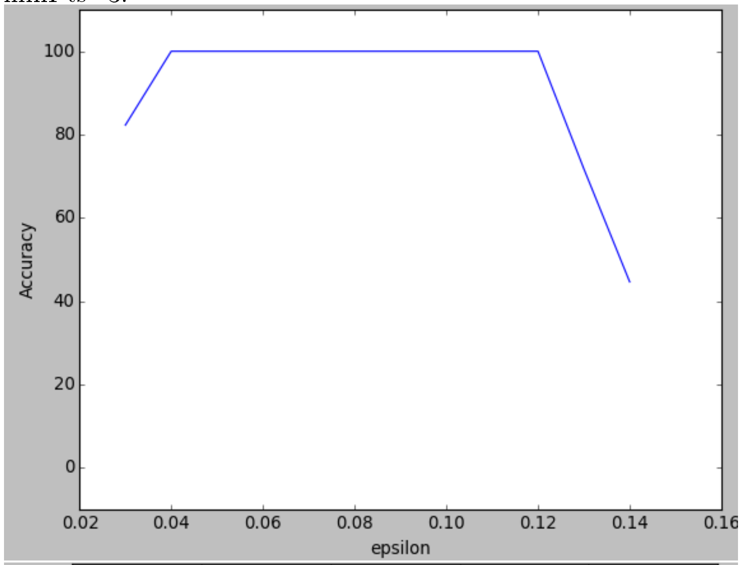
	Average Purity	Accuracy
Single Link	78.97%	37%
Complete Link	77.94%	70.66%
Centroid	80.92%	73.33%
Mean	78.56%	70.0%
Average	80.06%	73.0%
AdjComplete	76.01%	64.0%
Ward	75.333%	81.31%
NBDJoining	36.66%	36.66%

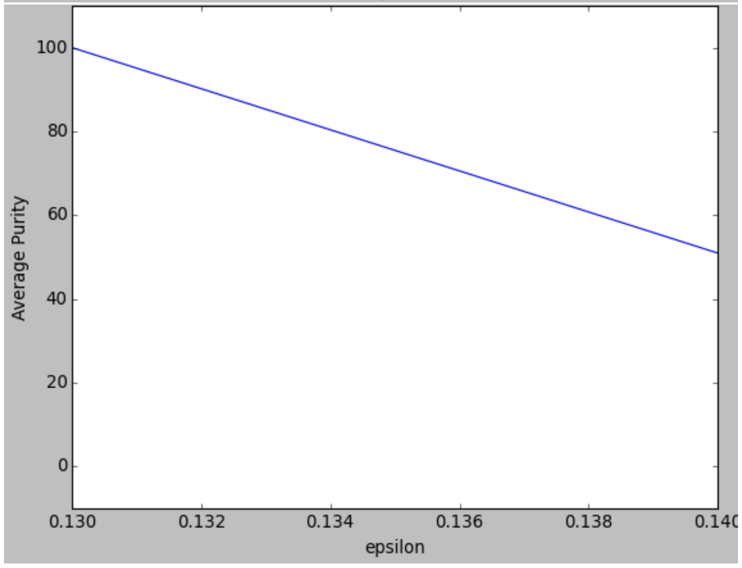
,

SPIRAL:

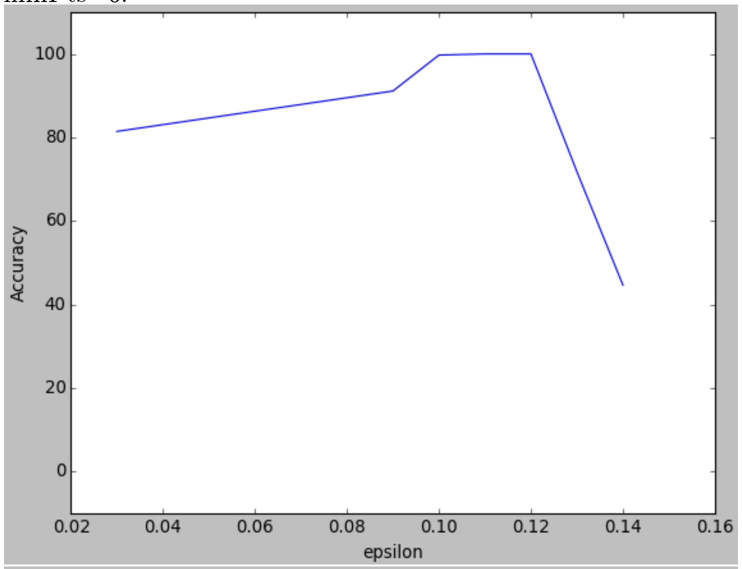
DBSCAN: DBSCAN recovers all the clusters perfectly. As can be seen, for epsilon=0.04 and minPts=3, we can fully recover the clustering with almost 100% accuracy.

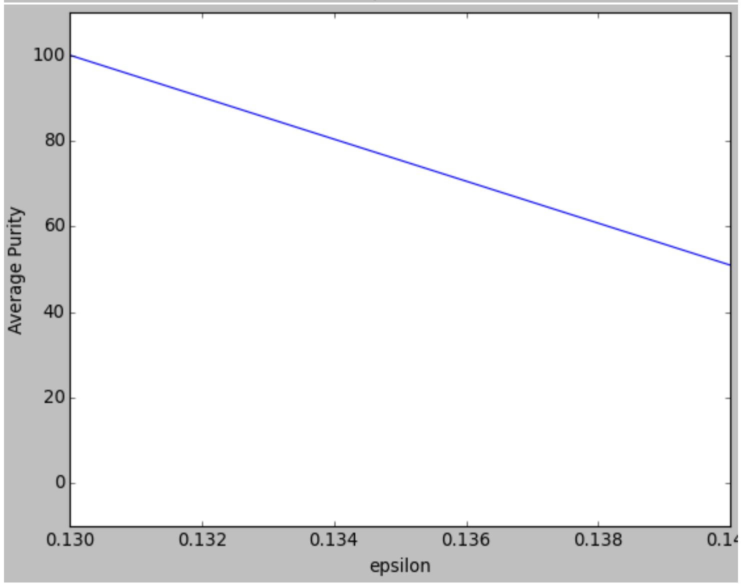
minPts=3:



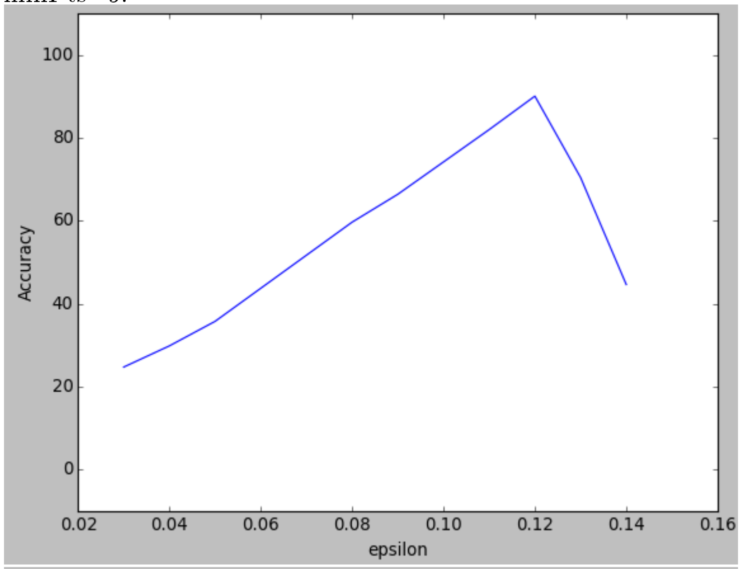


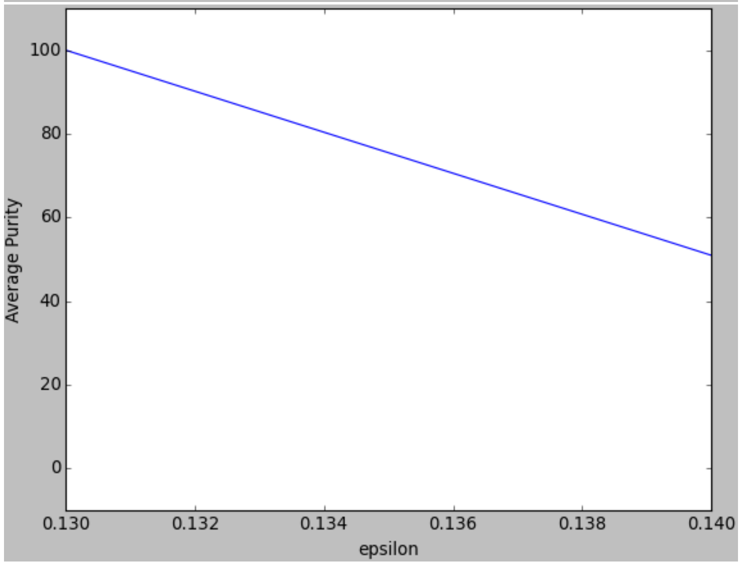
minPts=6:


Accuracy
100
80
60
40
20
0
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
epsilon


Average Purity
100
80
60
40
20
0
0.130
0.132
0.134
0.136
0.138
0.14
epsilon

minPts=9:


Accuracy
epsilon
100
80
60
40
20
0
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16


Average Purity
epsilon
100
80
60
40
20
0
0.130
0.132
0.134
0.136
0.138
0.140

Heirarchical clustering: Single link clustering gives the best performance, with 100% overall accuracy and purity in each cluster.

	Average Purity	Accuracy
Single Link	100%	100%
Complete Link	39.34%	38.14%
Centroid	42.51%	40.38%
Mean	39.60%	39.01%
Average	36.70%	36.22%
AdjComplete	78.10%	35.58%
Ward	44.30%	40.064%
NBDJoining	33.97%	33.97%

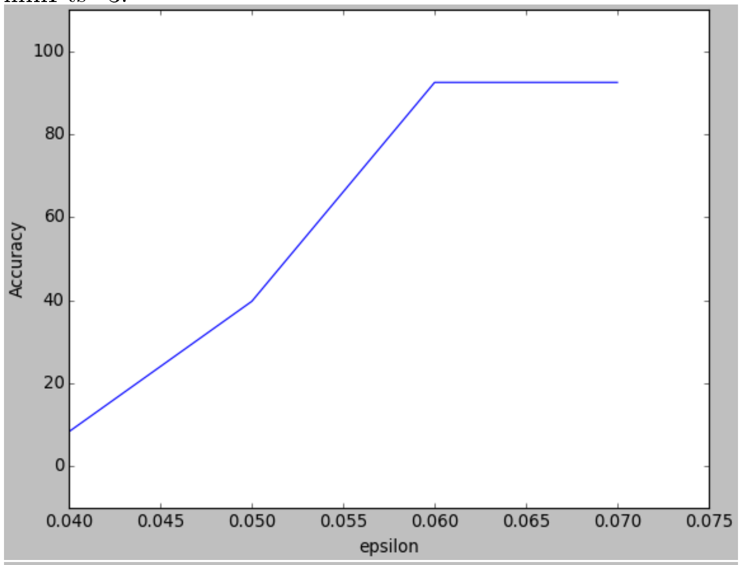
,

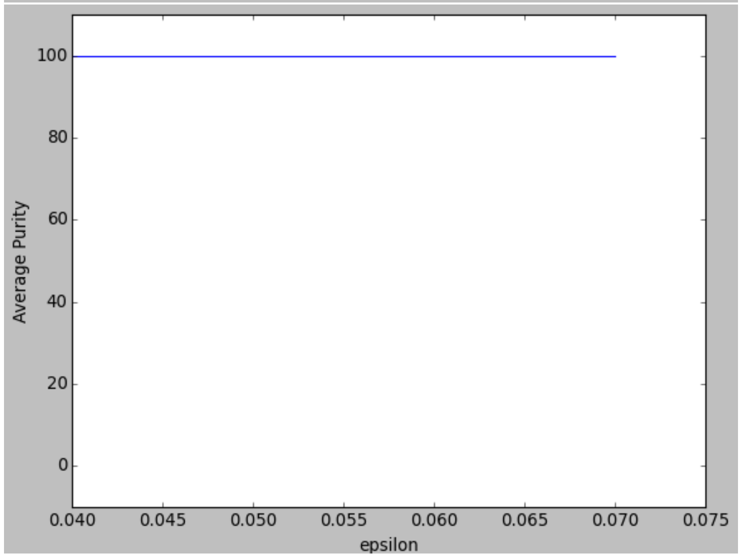
FLAME:

DBSCAN: The followingcharacteristics at m=1, 3, 6, 9 are observed:

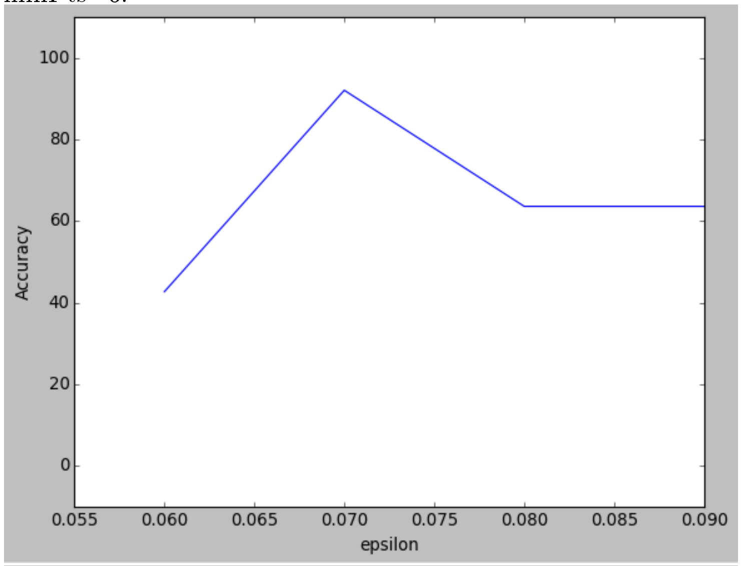
'

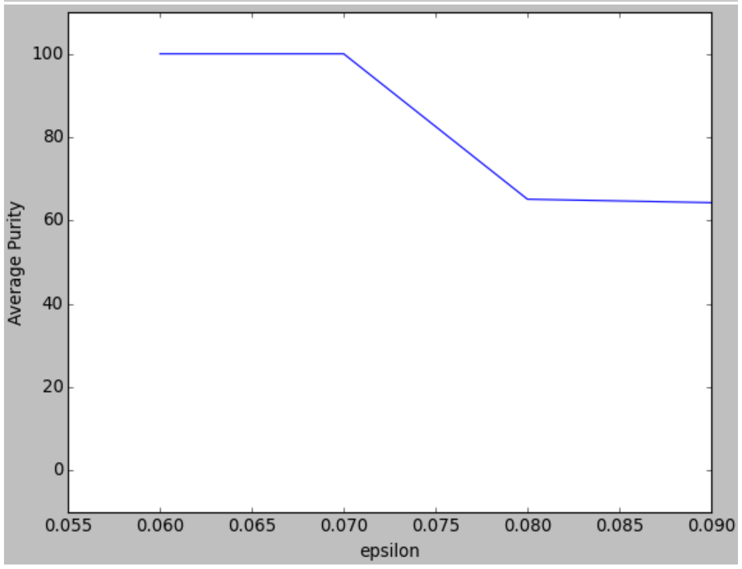
minPts=3:



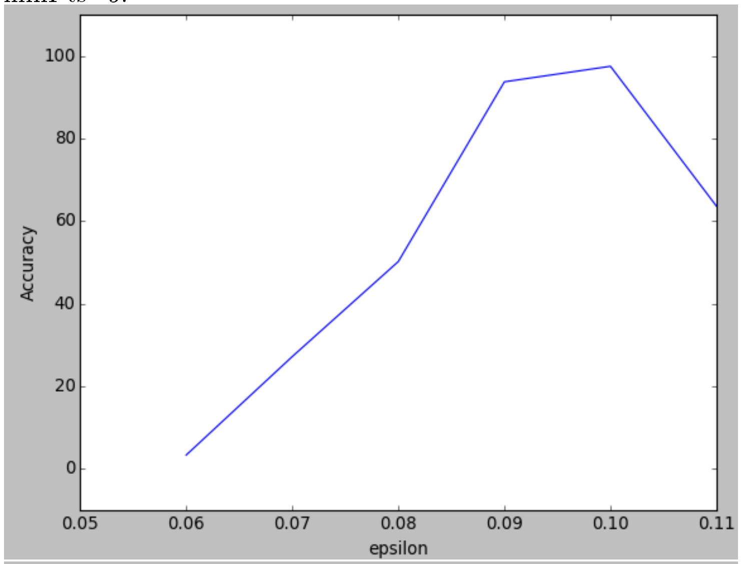


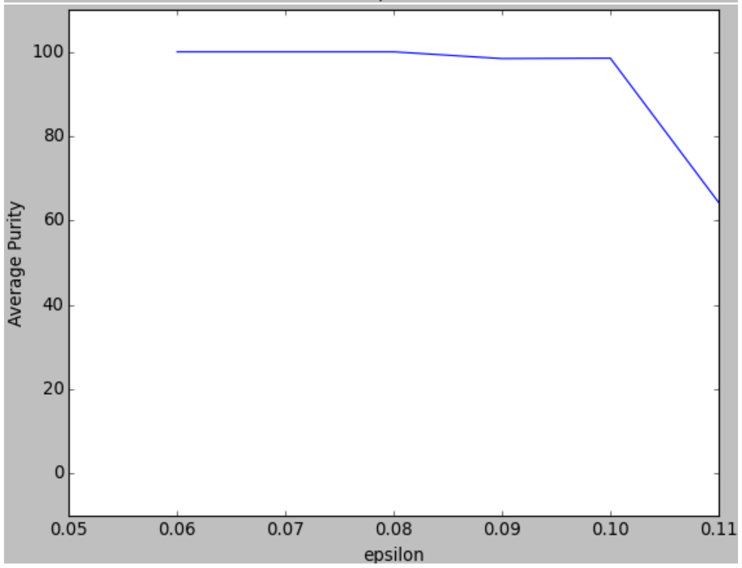
minPts=6:


Accuracy
100
80
60
40
20
0
0.055
0.060
0.065
0.070
0.075
0.080
0.085
0.090
epsilon


Average Purity
100
80
60
40
20
0
0.055
0.060
0.065
0.070
0.075
0.080
0.085
0.090
epsilon

minPts=9:


Accuracy
epsilon
0.05
0.06
0.07
0.08
0.09
0.10
0.11
0
20
40
60
80
100


Average Purity
epsilon
0.05
0.06
0.07
0.08
0.09
0.10
0.11
0
20
40
60
80
100

Heirarchical Clustering: Ward distance measure gave the best clustering accuracy and purity for this dataset.

	Average Purity	Accuracy
Single Link	82.14%	64.58%
Complete Link	76.44%	51.67%
Centroid	82.14%	64.58%
Mean	90.98%	92.08%
Average	84.25%	83.33%
AdjComplete	82.01%	64.17%
Ward	100%	100%
NBDJoining	63.75%	63.75%

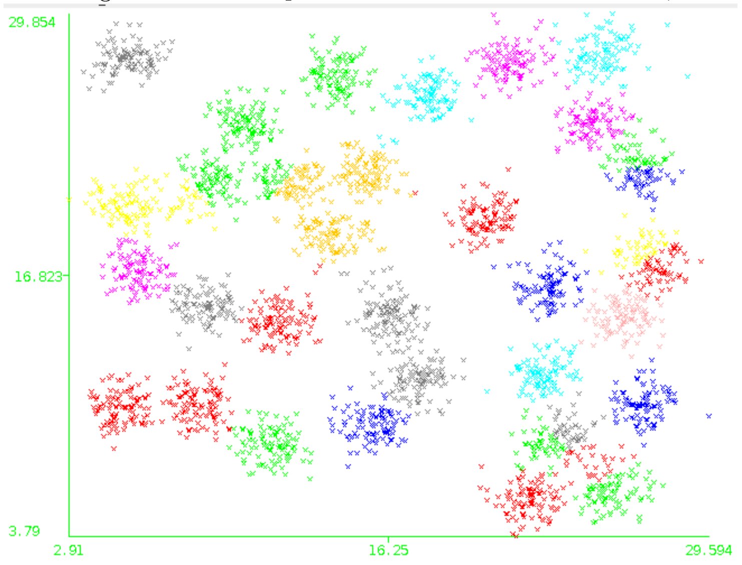
,
,
,

DBSCAN worked best for the PATHBASED dataset. In the spiral case, DBSCAN and single link are able to recover the clusters fulle. for FLAME, WARD is giving a 100% accuracy, and DBSCAN erforms reasonably well.

6 Clustering on D31 dataset:

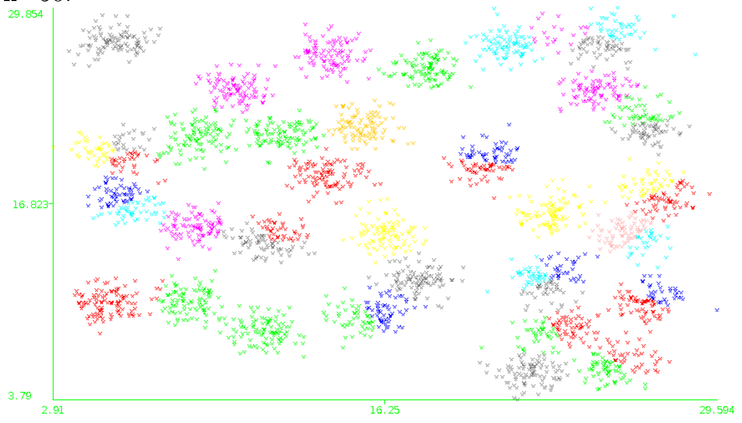
KMeans:

KMeans with 32 clusters over the large dataset was unable to run in weka with evaluation based on the class labels. However, it was able to output clusters over training data. The output clusters are shown as follows, with k=32:

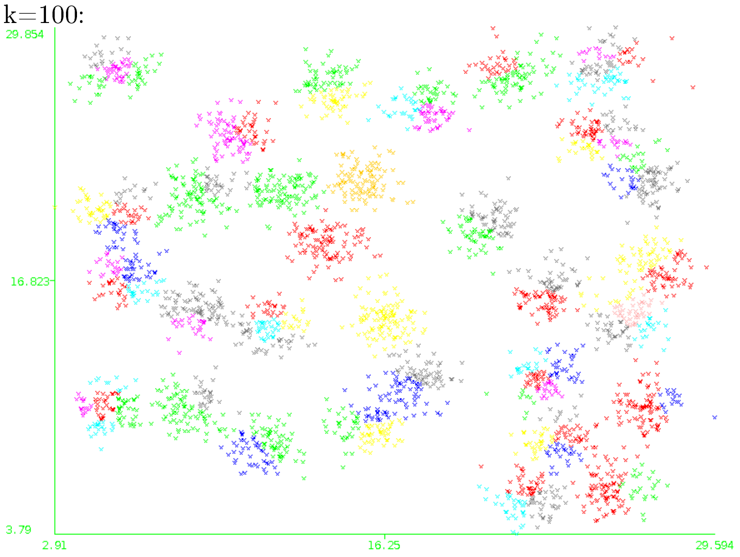


As we can see, some of the clusters are merged together and all clusters don't get recovered well. on increasing k to 50, then to 100, we the classes do get separated, but each of the classes get split. At k=50, we can make out the actual clusters involved, and some of the clusters are split further as a result of a large k.

k=50:



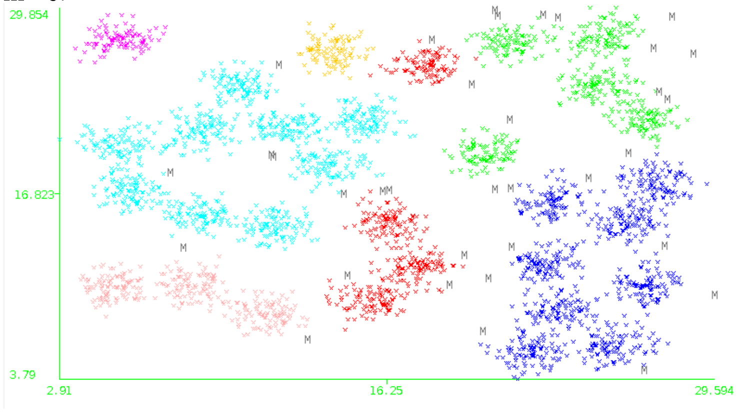
k=100:



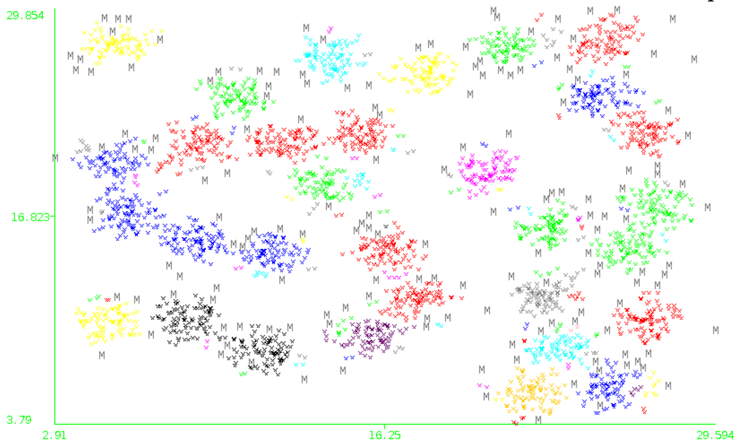
,

DBSCAN:On running DBSCAN with m=6, the clusters tend to be much bigger as the density. At the lowest epsilon giving fewer outliers (Eps=0.03), at m=6, The clustering is as follows:

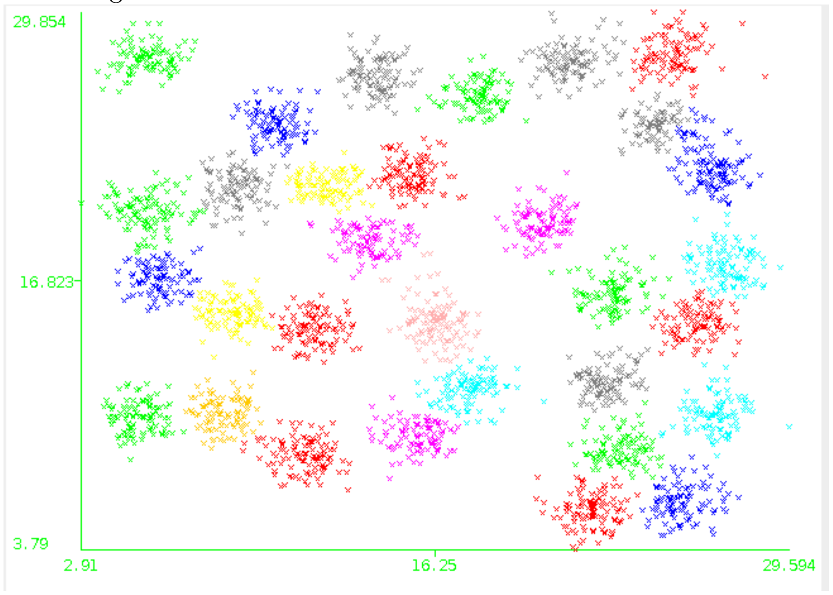
m=6:



At m=2 and Eps= 0.015(chosen such that we get few outliers) we get the following asignment. This is the best one can recover with DBSCAN, as sone of the clusters are connexted a narrow but dense stretch of datapoints.



Heirarchical clustering with Ward's linkage: Ward's linkage fully recovers all the clusters. As with KMeans, it is unable to do class evaluation, but only basedoff of visualization, we can see that all clusters are fully recovered using Ward's linkage.



,