Name: Garvit Pugalia
UID: 504628127

<u>Homework 4</u>

Question 1: Clustering Evaluation

For purity, we need to first create a cross-referencing matrix:

Ground Truth Label	**Algorithm Output Label**				**Total**
	1	**2**	**3**	**4**	
1	0	1	4	0	**5**
2	5	0	0	0	**5**
3	0	5	0	0	**5**
4	0	0	1	4	**5**
Total	**5**	**6**	**5**	**4**	**20**

Now, we can find the largest cluster matching each ground truth class (maximum value in each row) to get:

Purity = (4 + 5 + 5 + 4) / 20 = **0.9**

From the cross-referencing matrix above, we can also get the following probabilities:

$P(w_1 \wedge C_2) = 1/20$	$P(w_1) = 5/20$	$P(C_1) = 5/20$
$P(w_1 \wedge C_3) = 4/20$	$P(w_2) = 5/20$	$P(C_2) = 6/20$
$P(w_2 \wedge C_1) = 5/20$	$P(w_3) = 5/20$	$P(C_3) = 5/20$
$P(w_3 \wedge C_2) = 5/20$	$P(w_4) = 5/20$	$P(C_4) = 4/20$
$P(w_4 \wedge C_3) = 1/20$		
$P(w_4 \wedge C_4) = 4/20$		

Now, we can compute the components that make up the NMI formula:

$$H(\Omega) = -0.25 \log(0.25) - 0.25 \log(0.25) - 0.25 \log(0.25) - 0.25 \log(0.25)$$
$$= -\log(0.25) = 2.0$$

Name: Garvit Pugalia
UID: 504628127

$$H(C) = -0.25 \log(0.25) - 0.3 \log(0.3) - 0.25 \log(0.25) - 0.2 \log(0.2)$$
$$= -0.5 \log(0.25) - 0.3 \log(0.3) - 0.2 \log(0.2)$$
$$= 1 + 0.52108967825 + 0.46438561897$$
$$= 1.98547529723$$

Similarly, we can compute the information gain using the formula and the probabilities of each intersection of cluster and ground truth labels:

$$I(\Omega, C) = 0.05 \log(0.05 / 0.25 \times 0.3) + 0.2 \log(0.2 / 0.25 \times 0.25) + 0.25 \log(0.25 / 0.25 \times 0.25) + 0.25 \log(0.25 / 0.25 \times 0.3) + 0.05 \log(0.05 / 0.25 \times 0.25) + 0.2 \log(0.2 / 0.25 \times 0.2)$$
$$= - 0.02925 + 0.33561 + 0.5 + 0.43424 - 0.016096 + 0.4$$
$$= 1.6245$$

Now, we can calculate the normalized mutual information (NMI) as:

$$NMI = I(\Omega, C) / \sqrt{H(C)H(\Omega)}$$
$$= 1.6245 / \sqrt{2.0 \times 1.98547529723}$$
$$= \mathbf{0.8152}$$

For precision, recall and F-measure, we need to create a confusion matrix, where we count the (mis)clustering of similar elements and the (mis)clustering of dissimilar elements. Therefore, we need to compute the following for each *pair* of datapoints:

Ground Truth Label	**Algorithm Output Label**	
	Same	**Different**
Same	TP = 32	FN = 8
Different	FP = 9	TN = 141

Therefore, we get:

$$\text{Precision} = TP / (TP + FP) = 32/41 = \mathbf{0.780}$$
$$\text{Recall} = TP / (TP + FN) = 32/40 = \mathbf{0.8}$$

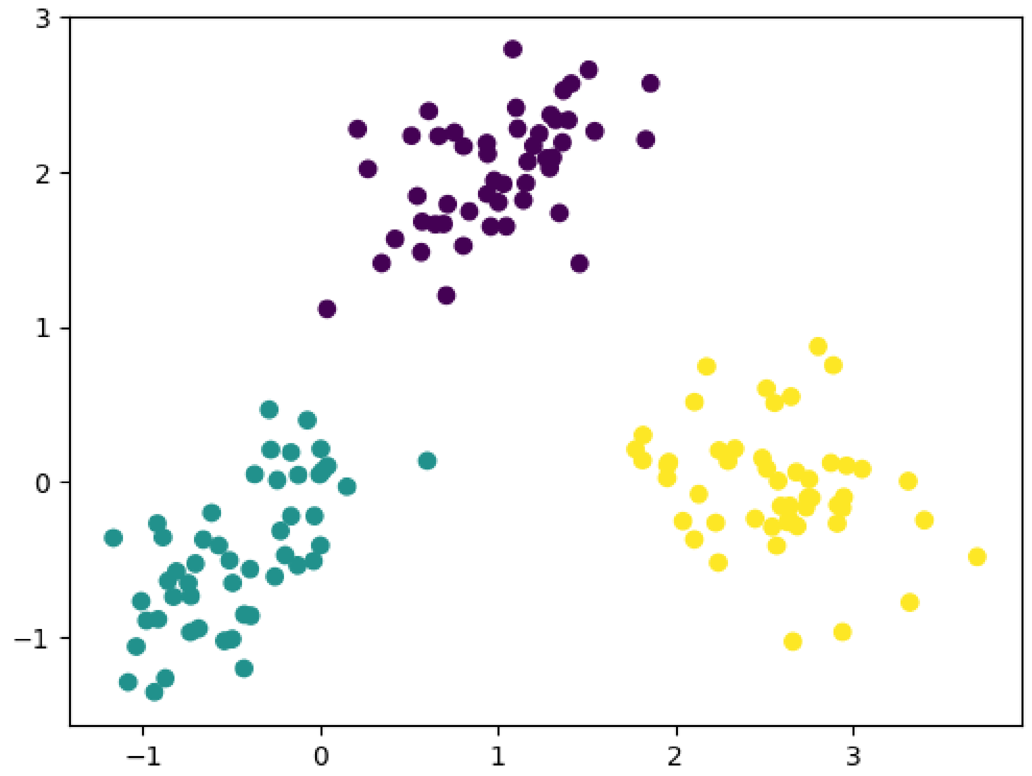
$$\text{F-measure} = 2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$$
$$= 1.25878 / 1.580 = \mathbf{0.790}$$

Name: Garvit Pugalia
UID: 504628127

Question 2: K-Means

(1) The corresponding lines are filled in the Kmeans.py file, where the new point is added to the closest cluster measured in Euclidean distance.

(2) Dataset 1:
After 3 iterations, we have:

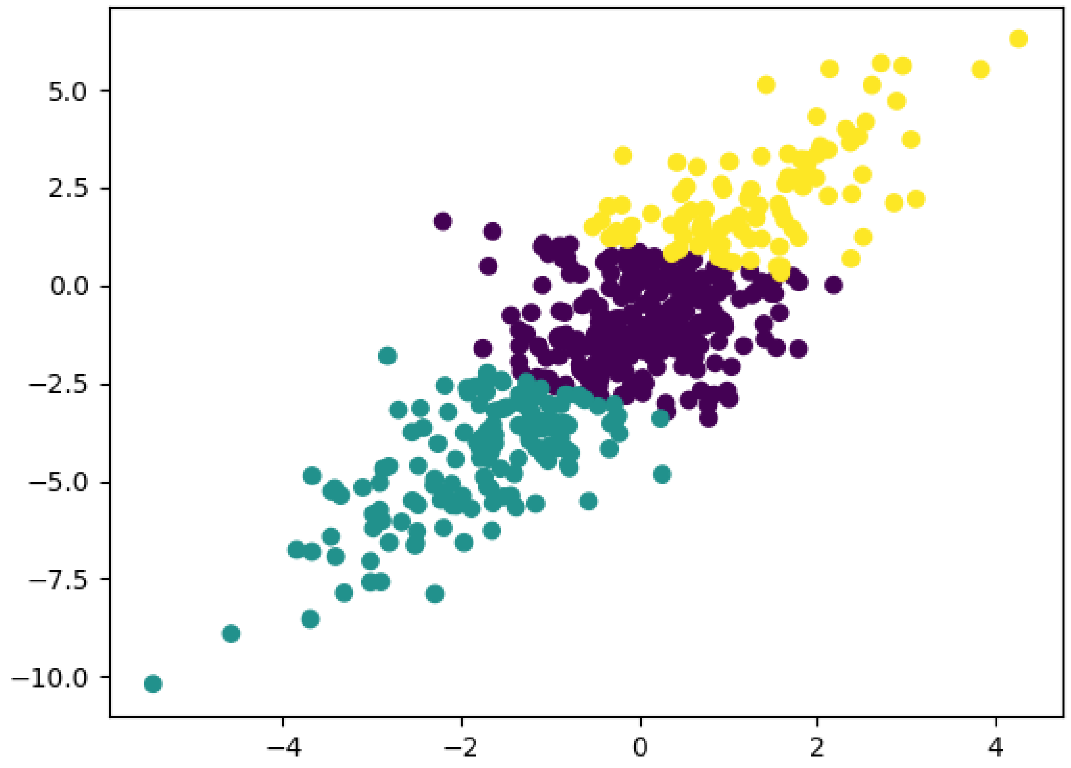


Purity = 1.0
NMI = 1.0

Since the purity and NMI are both 1.0, the clustering matches each datapoint with the correct cluster.

Name: Garvit Pugalia
UID: 504628127

Dataset 2:
After 8 iterations, we have:

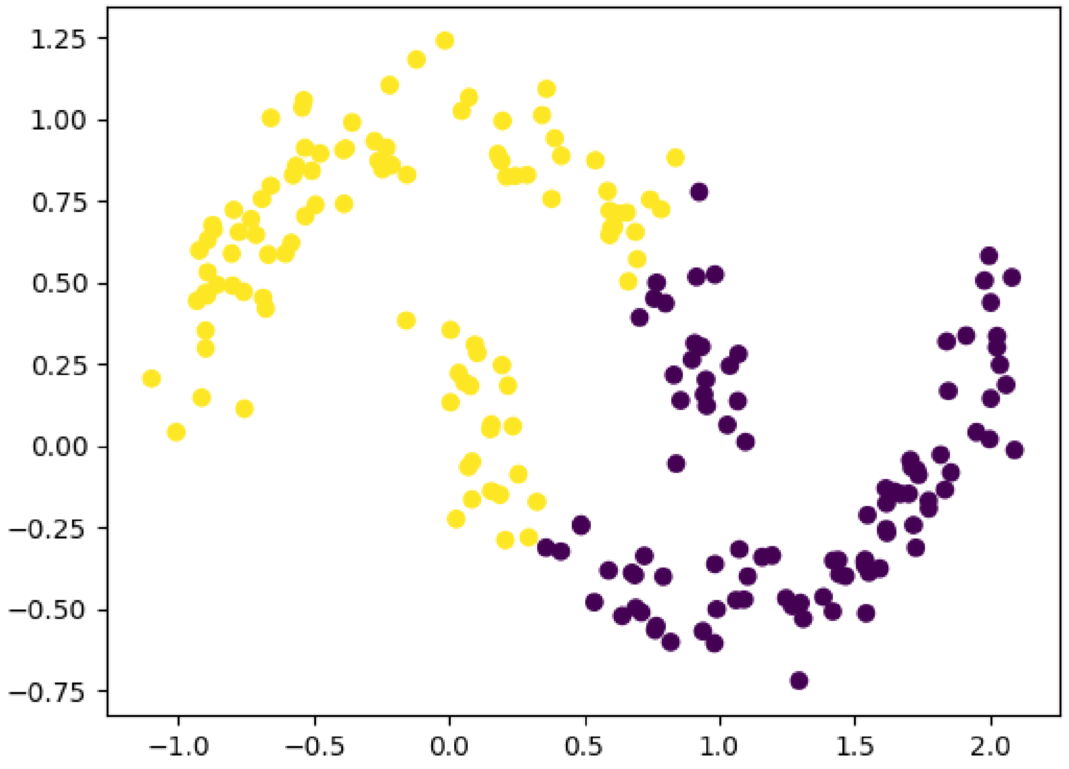


Purity = 0.764
NMI = 0.0617079727946

While the purity is close to 1.0, the NMI is very low. We can observe that the dataset is very dense in the above graph. Since purity doesn't consider the tradeoff of creating more clusters, it can provide an incorrect image. The low NMI can mean that using fewer clusters could provide a better model.

Name: Garvit Pugalia
UID: 504628127

Dataset 3:
After 6 iterations, we have:



Purity = 0.78
NMI = 0.169704955284

This situation is similar to Dataset 2, where the Purity is high but the NMI is low. Clearly, in the graph, the two parabolas are not clustered together. Since K-Means is a weak clustering algorithm with non-spherical cluster shapes, it is unable to properly classify the dataset, and the NMI is low.

Name: Garvit Pugalia
UID: 504628127

(3) K-means was the quickest algorithm out of the three clustering methods. It runs in O(tkn) time, where n is number of objects, k is number of clusters, and t is number of iterations. Therefore, since the algorithm used < 10 iterations and k < 4, we can declare that K-means ran in O(n).

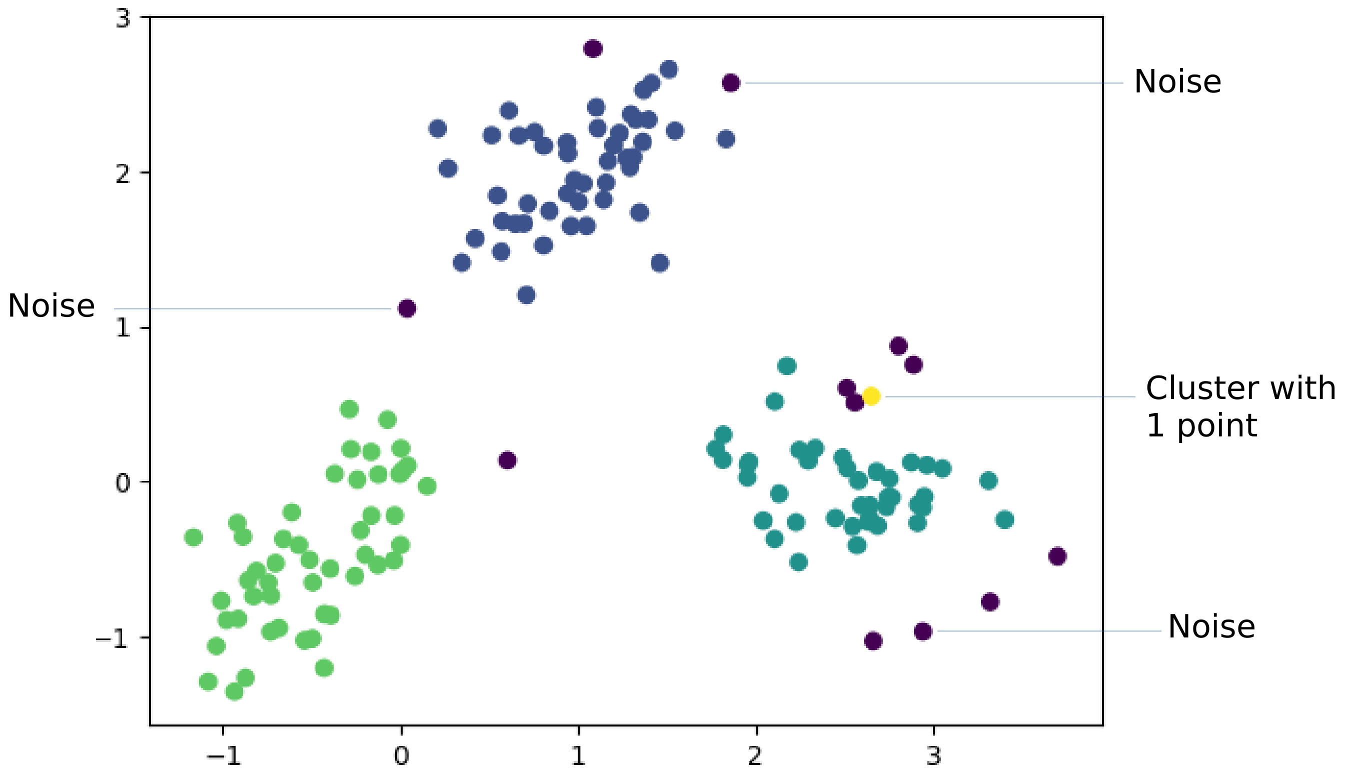
A major disadvantage of K-means is that the number of clusters, k, is specified in advance. It is possible that the clustering works better with a higher k, however, K-means cannot identify this. Also, from Dataset 3, it is clear that the algorithm doesn’t work well with non-spherical cluster shapes. Finally, the algorithm is sensitive to noise, and is unable to identify outliers like the other approaches.

Name: Garvit Pugalia
UID: 504628127

Question 3: DBSCAN

(1) The corresponding lines are filled in the DBSCAN.py file, where a directly density reachable point is added to the current cluster. As an after-effect, the isAssignedToCluster attribute is set to True.

(2) Dataset 1:
Esp = 0.356083270505
Noise points = 12



Purity = 0.92
NMI = 0.984741919676

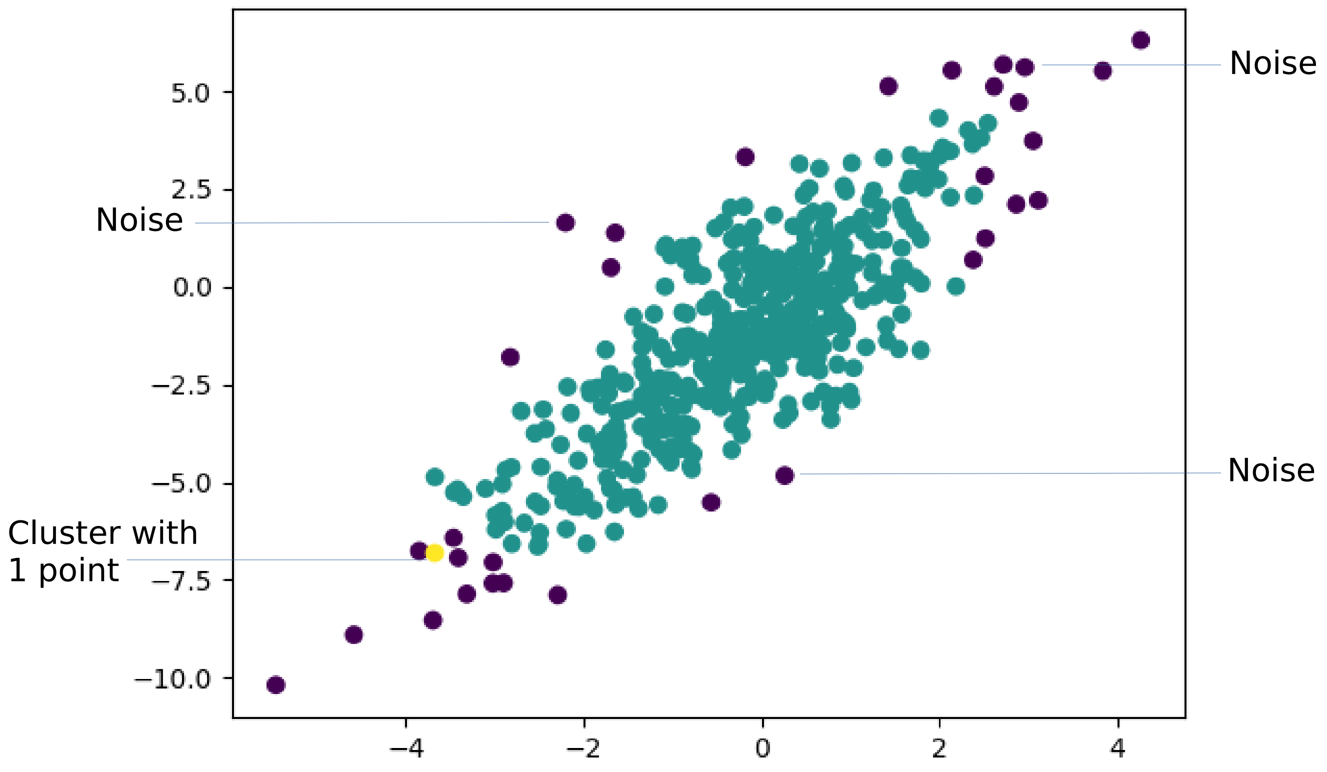
The values of Purity and NMI are close to 1.0, which means the clustering was successful. However, the K-means algorithm performed better on the dataset. This could be because the dataset has low density and the hyperparameters aren't adjusted correspondingly (Esp is too low). This leads to overclassification of noise points, and a low NMI and Purity.

Name: Garvit Pugalia
UID: 504628127

Dataset 2

Esp = 0.465682418877

Noise points = 32



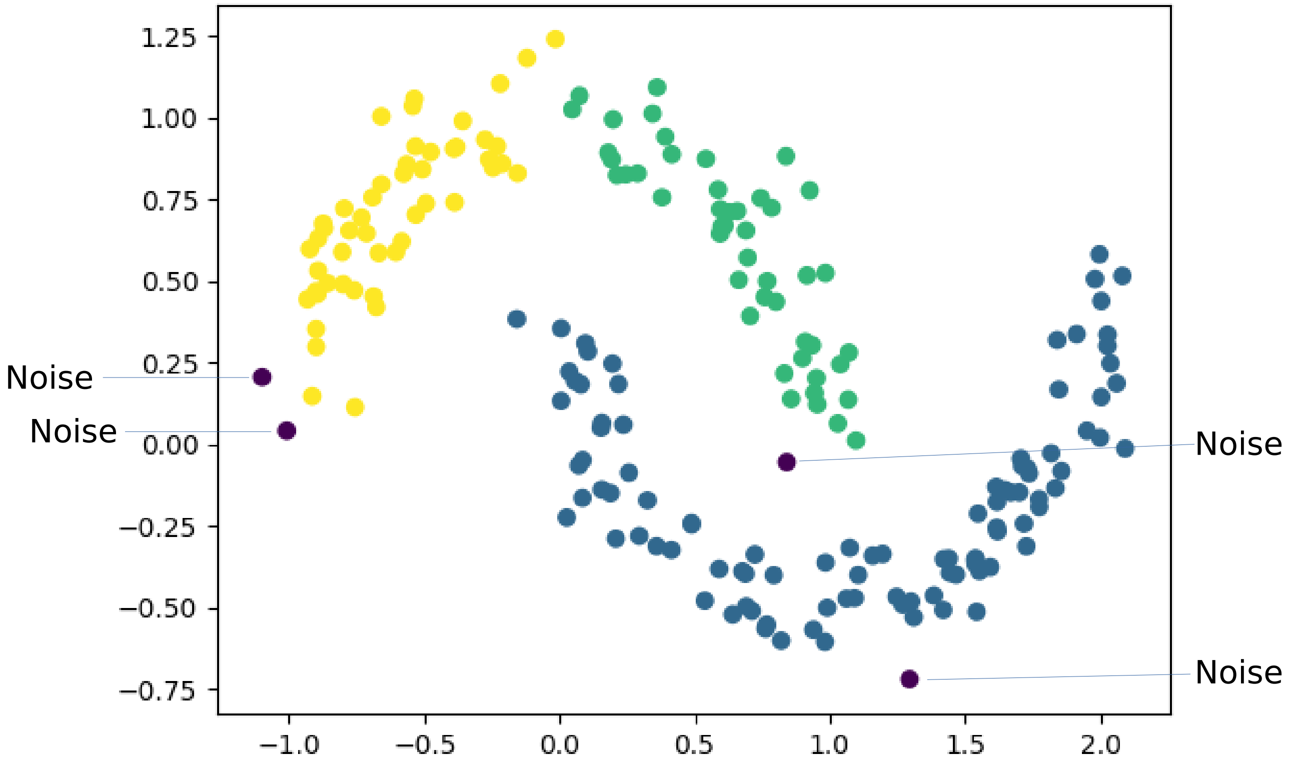
Purity = 0.708

NMI = 0.00525345087768

The Purity is high since most classes will have a large intersection with the large cluster. However, the NMI is very low. This is because the dataset is incredibly dense, and the hyperparameter is not adjusted properly (Esp is too high). Therefore, most points are considered as part of one huge cluster, with most of the rest of the points marked as noise. Since the entropy of the clusters, and entropy of classes, is high, the NMI comes out close to zero.

Name: Garvit Pugalia
UID: 504628127

<u>Dataset 3</u>
Esp = 0.186520964767
Noise points = 4



Purity = 0.98
NMI = 0.817972012066

The Purity and NMI are close to 1, which means the clustering was successful. The NMI is much larger than the NMI with K-means. The DBSCAN algorithm works based on density. Therefore, if the hyperparameters are correctly set, the approach can recognize differently shaped clusters, and can obtain a better NMI than K-means.

Name: Garvit Pugalia
UID: 504628127

(3) DBSCAN can work better than K-means since it performs clustering with a variable number of clusters. Therefore, it can find the best clustering while considering different numbers for k. In all of the previous graphs, we observe that DBSCAN is able to identify and ignore outliers, which is not possible with K-means. Also, since each datapoint is scanned once, the algorithm can be faster than K-means as n increases. Finally, as stated in the results of Dataset 3, DBSCAN can recognize clusters of many different shapes.

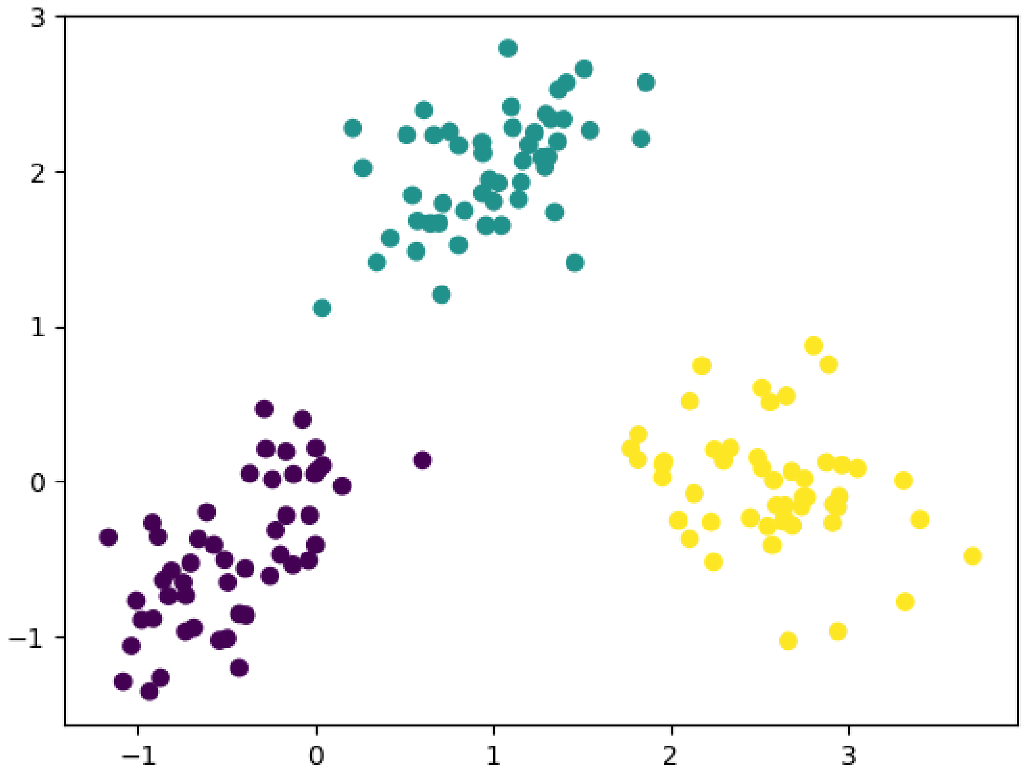
There are still a few setbacks with using DBSCAN. With K-means, the only hyperparameter is the number of clusters. In our approach, the Eps (environment radius) and MinPts (minimum number of points in environment) are two hyperparameters used by DBSCAN. As seen in the results of Dataset 1 and Dataset 2, the NMI is low and the clustering is poor due to incorrectly computed hyperparameters.

Name: Garvit Pugalia
UID: 504628127

Question 4: GMM

(1) The corresponding lines in DataPoints.py and GMM.py were filled in. In DataPoints, the functions to compute mean, standard deviation, and the covariance matrix for clusters were completed. In GMM, the weights for each <datapoint, class> pair were converted into probabilities. The cov(X,Y) is updated appropriately (according to formula from lecture), and the w[j] i.e. the sum of all weights with class = j, was computed. These are then used to perform the EM algorithm and divide the points into clusters.

(2) Dataset 1:
After 23 iterations, we have:

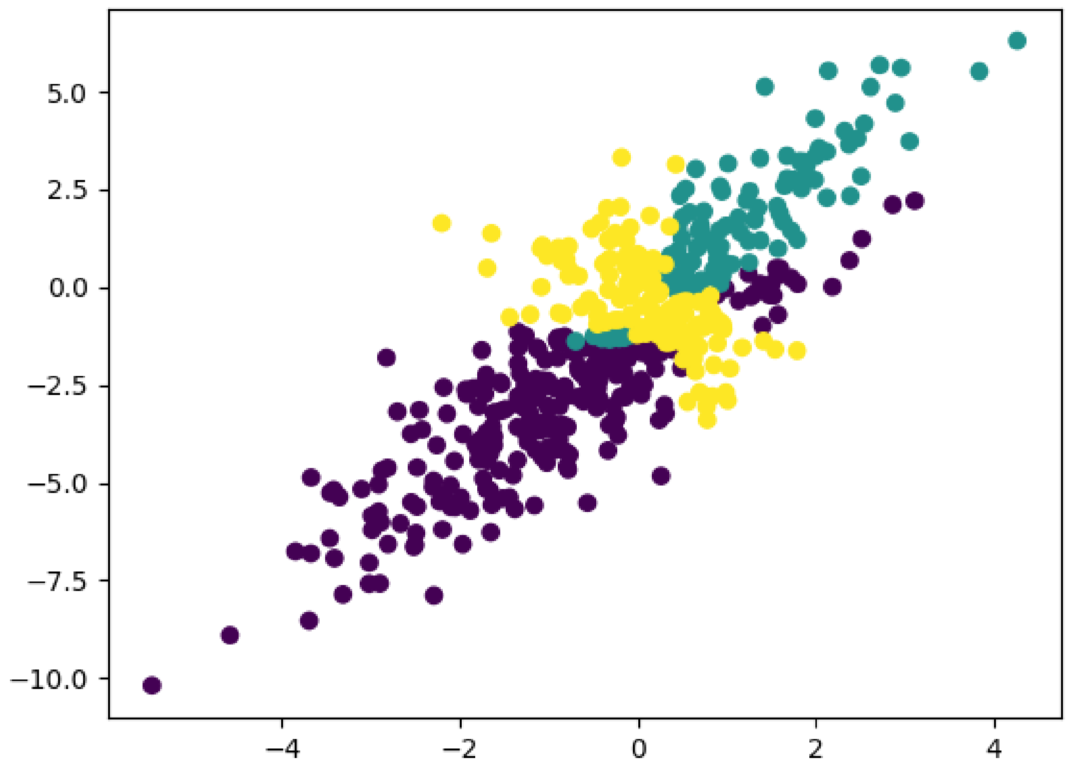


Purity = 1.0
NMI = 1.0

Since the purity and NMI are both 1.0, the clustering matches each datapoint with the correct cluster.

Name: Garvit Pugalia
UID: 504628127

Dataset 2:
After 87 iterations, we have:

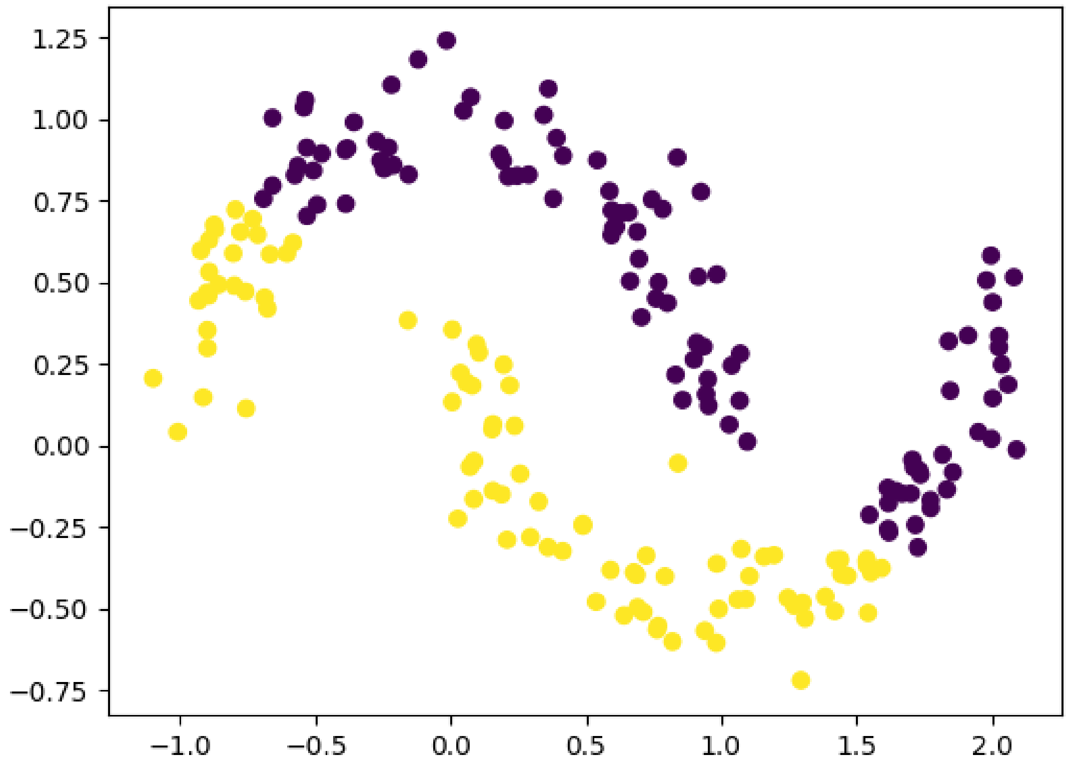


Purity = 0.764
NMI = 0.0777649722048

Both Purity and NMI are comparable to the respective values from K-means and DBSCAN. This is surprising as GMM is able to deal with clusters of different densities (where DBSCAN failed). However, like K-means, the number of clusters is hard to estimate. This, along with the possibility of convergence to a local minima, could be the reason for a lower NMI.

Name: Garvit Pugalia
UID: 504628127

Dataset 3:
After 90 iterations, we have:



Purity = 0.69
NMI = 0.075947839504

This situation is similar to K-means, where the Purity is high but the NMI is low. Clearly, in the graph, the two parabolas are not clustered together. Like K-means, GMM is a weak clustering algorithm with non-spherical cluster shape, therefore the NMI is low.

Name: Garvit Pugalia
UID: 504628127

(3) With GMM, the number of parameters are minimum. This is a big advantage over the other two approaches, as clustering is very sensitive to parameters in general. The algorithm is closely related to underlying generative models (statistical models that could have produced the dataset), which can drastically improve accuracy. Also, the approach works better than DBSCAN and K-means when the dataset contains clusters of varying sizes and densities.

Like K-means, GMM falls short with certain datasets. For example, the algorithm works poorly with non-spherical cluster shapes, which was clearly seen in the results of Dataset 3. Unlike K-means, it is possible that the GMM algorithm reaches a local optima (as suggested in the results of Dataset 2). Therefore, there is a factor of randomness involved which can be handled through more iterations. Finally, after running all three algorithms, I observed that GMM runs much slower than the two competitors. With a larger dataset, the computational cost of GMM increases faster than DBSCAN and K-means.