Michael Wang
CS 145 HW 4

1) Output clusters 1,2,3,4 contain [2,2,2,2,2], [3,3,3,3,3,2], [1,1,1,1,4], and [4,4,4,4] respectively. Running a python program shows that the number of true/false positives and true/false negatives are 32, 9, 141, and 8 respectively.

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

Precision: = TP/(TP+FP) = 32/(32+9) = 0.780

Recall: = TP/(TP+FN) = 32/(32+8) = 0.8

F-measure: 2*Precision*Recall/(Precision+Recall) = 2*0.780*0.8/(0.780+0.8) = 0.790

Normalized Mutual Information:

I = 5/20log(20*5/(5*6)) + [5/20log(20*5/(6*5)) + 1/20log(20*1/(6*6))] + [5/20log(20*4/(5*4)) + 1/20log(20*1/(5*4))] + 4/20log(20*4/(4*5)) = 1.726

H(C) = -(5/20log(5/20) + 6/20(log6/20) + 5/20log(5/20) + 4/20log(4/20)) = 1.985

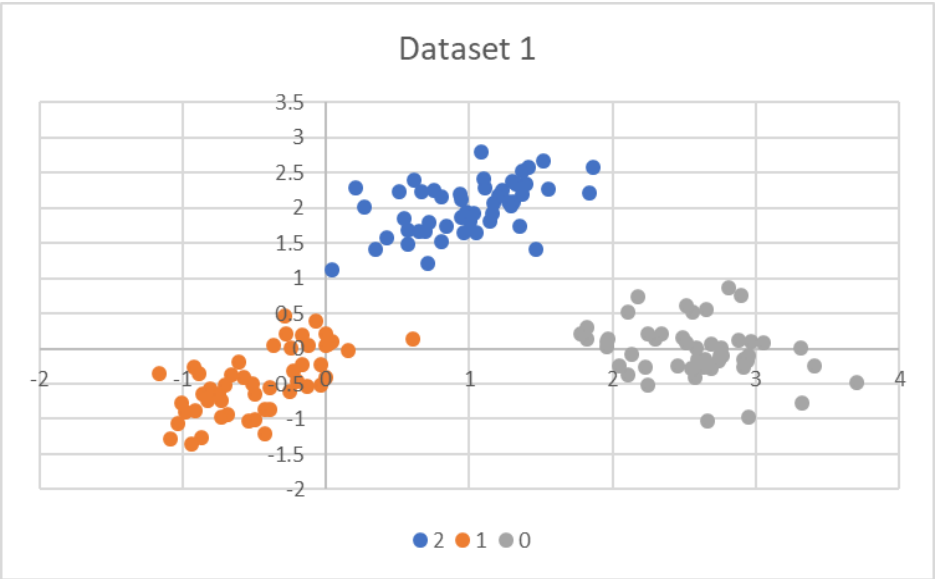
H(Ω) = -(4/20log(4/20) + 6/20log(6/20) + 5/20log(5/20) + 5/20log(5/20)) = 1.985

NMI = I/sqrt(H(C)H(Ω)) = 1.726/1.985 = 0.870

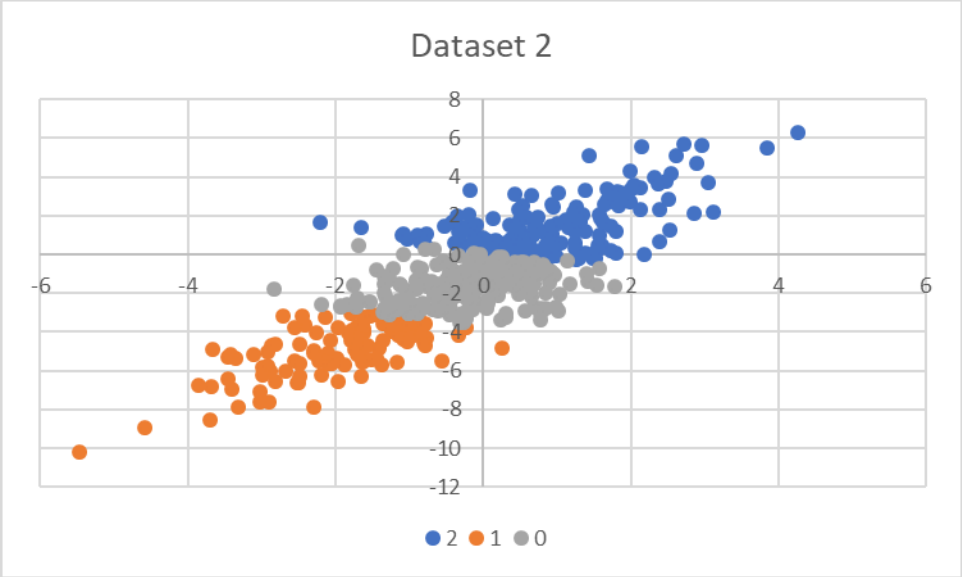
2)

K-Means

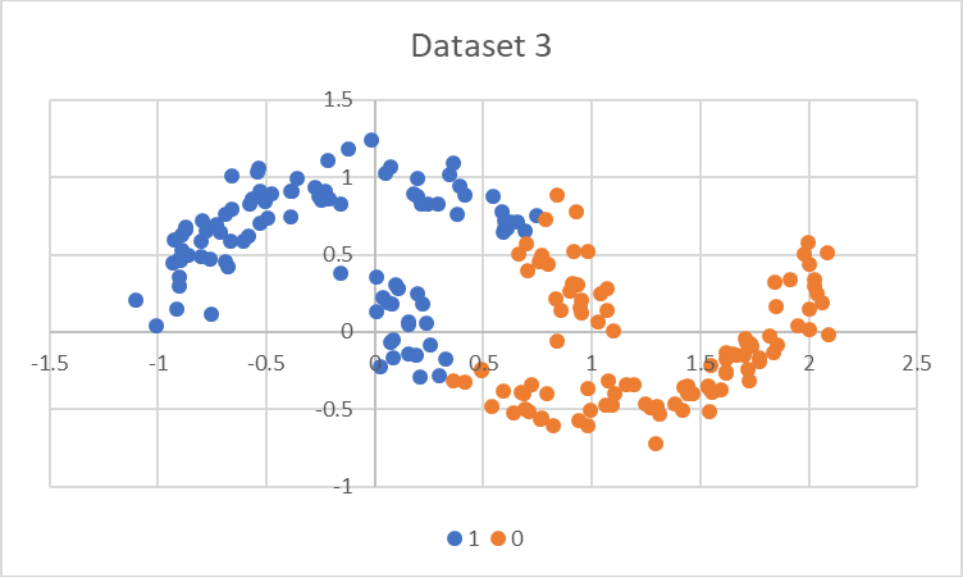
Iteration :3
Purity is :1.0
NMI :1.0
Cluster 0 size :50
Cluster 1 size :50
Cluster 2 size :50



Iteration :6
Purity is :0.764
NMI :0.0468513659549
Cluster 0 size :230
Cluster 1 size :108
Cluster 2 size :162



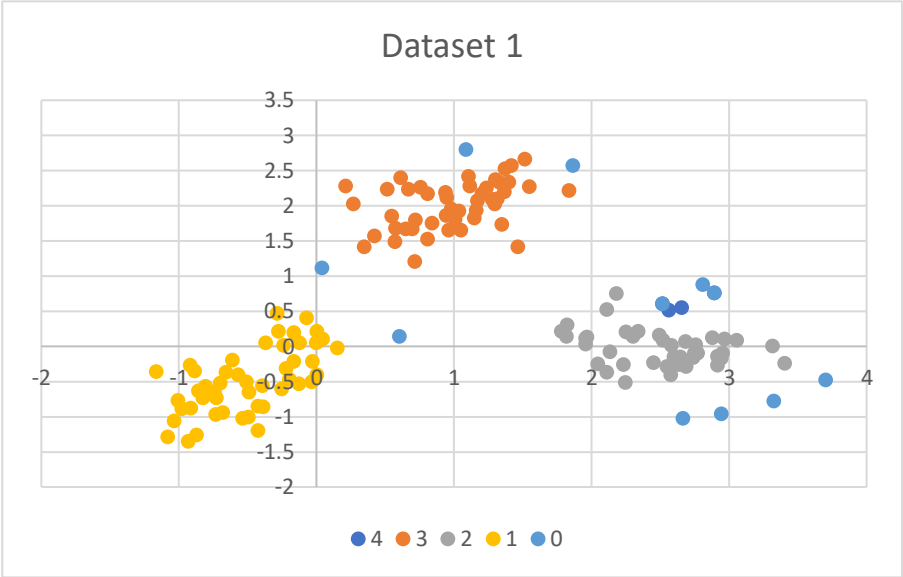
Iteration :4
Purity is :0.76
NMI :0.14502499937
Cluster 0 size :102
Cluster 1 size :98



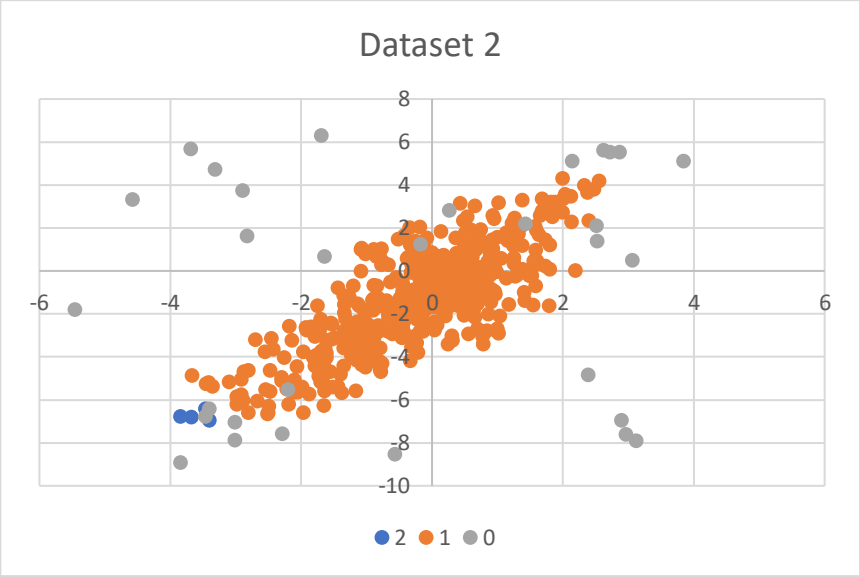
The primary strength of K-means is its relative simplicity and by extension the efficiency of its algorithm which is linear on the number of examples. That's mostly where the good news for K-means ends as it suffers from several weaknesses that the other two algorithms partially cover. The most obvious weakness is the need to specify k before running the algorithm. This may not be too bad if the correct k is known. However, the code given to us assumes k to be the number of labels which gets us into trouble with dataset 2 which contains 3 labels but all the datapoints clearly belong to a single cluster. As such, K-means does its best to split the cluster into 3 sub-clusters, but its best isn't good enough. Another weakness of K-means is that it doesn't identify clusters of varying densities and sizes too well. For example, a large non-dense cluster might have a lower cost when split into two while two smaller dense clusters are merged. This issue isn't apparent in the given datasets but is something to kept in mind. Another weakness of k means is that it is only suitable for spherical clusters and given clusters of different shape it will attempt to superimpose spheres onto them. This is demonstrated in dataset 3 where the two ground truth crescent clusters are essentially split down the middle and shared by the two output clusters.

DBSCAN

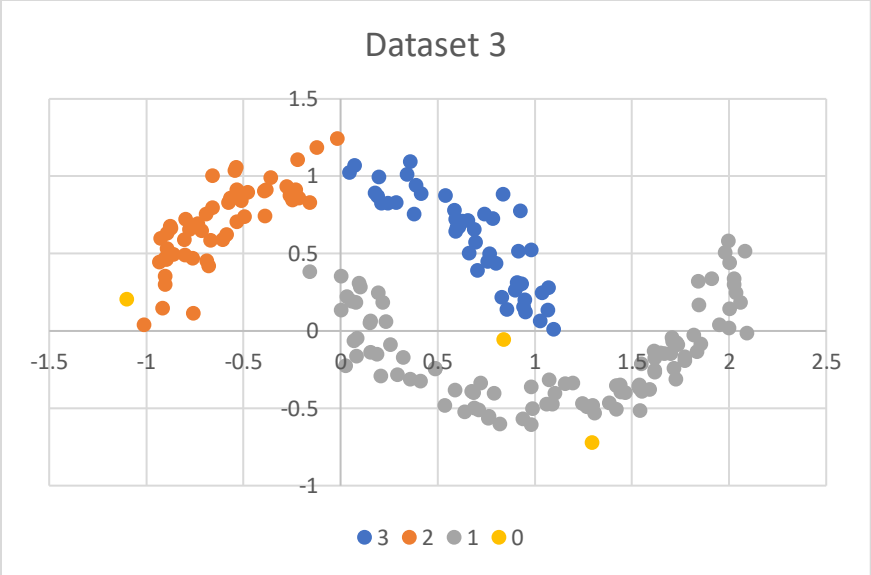
For dataset1
Esp:0.3560832705047313
Number of clusters formed :4
Noise points :11
Purity is :0.94
NMI :0.9590647490609898
Cluster 0 size :49
Cluster 1 size :41
Cluster 2 size :47
Cluster 3 size :4



For dataset2
Esp :0.4656824188773015
Number of clusters formed :2
Noise points :32
Purity is :0.714
NMI :0.011352036737124684
Cluster 0 size :467
Cluster 1 size :4



For dataset3
Esp :0.18652096476712493
Number of clusters formed :3
Noise points :3
Purity is :0.985
NMI :0.8173489274692755
Cluster 0 size :99
Cluster 1 size :51
Cluster 2 size :47



The strengths of DBSCAN are that it does not require the number of clusters to be specified before-hand and that it can separate clusters of arbitrary shape, nut just spherical ones. The first strength is best shown by dataset 2 in which DBSCAN placed the majority of points into a single cluster while the other two algorithms forcibly split it into three. Dataset 3 demonstrates the latter strength with DBSCAN successfully identifying the crescent shape of the clusters while the other two algorithms had to resort to drawing ovals over the clusters. Another strength of DBSCAN is its notion of noise which makes it resistant to outliers, although this characteristic works to its detriment in the given cases since the ground truths had no noise. The weaknesses of DBSCAN are that it is has difficulty with clusters of varying densities and that it requires eps and min parameters to be defined beforehand. The first weakness isn't very noticeable in the given datasets since they all the clusters had similar densities. The second weakness is more apparent – dataset3 would have likely been correctly split into two clusters rather than 3 had a larger eps been used.

Gaussian Mixture Model

For Dataset1

Number of Iterations = 23

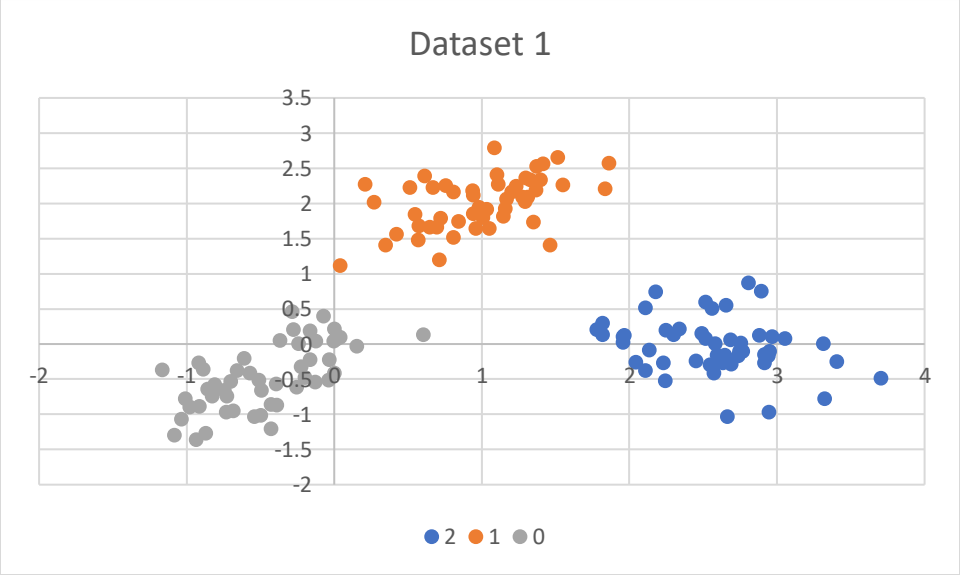
After Calculations
Final mean =
-0.46247229923787236 -0.4638730939529598
0.9898955265866389 2.011802031563348
2.5734296991089827 -0.02711297398490922

Final covariance =
For Cluster : 1
0.14918912639706094 0.1173470001849541
0.1173470001849541 0.2155513332586024

For Cluster : 2
0.16028152865011036 0.07486762680764208
0.07486762680764208 0.13939759162689855

For Cluster : 3
0.1803889474897513 -0.04672155381408792
-0.04672155381408792 0.15205897951248673

Purity is :1.0
NMI :1.0
Cluster 0 size :50
Cluster 1 size :50
Cluster 2 size :50



For dataset2

Number of Iterations = 82

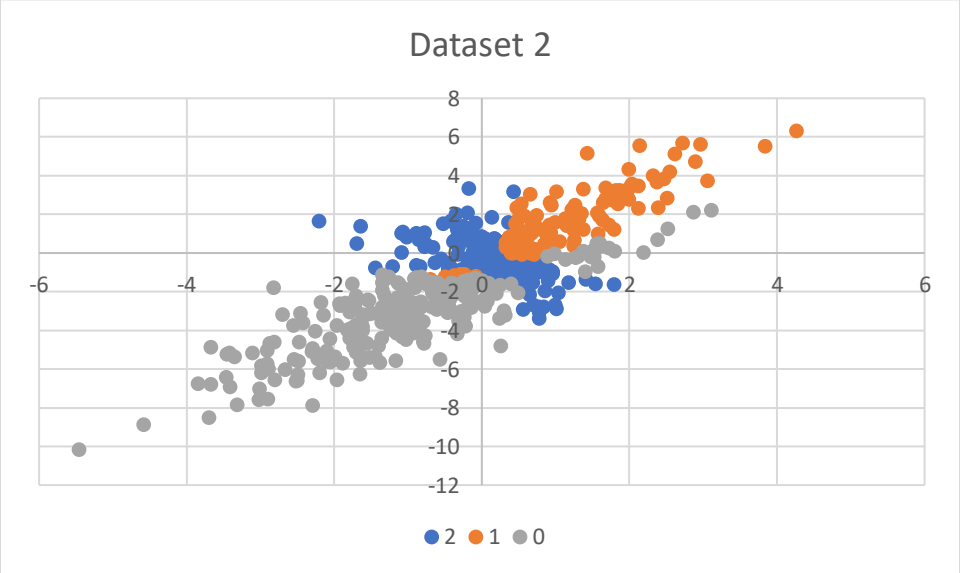
After Calculations
Final mean =
-0.7693449733277089 -2.700716957559499
0.22150686874538947 -0.32545481990106995
0.04901772949170768 -0.38076263808826816

Final covariance =
For Cluster : 1
2.177369466262199 2.987069709464784
2.987069709464784 5.635492171816077

For Cluster : 2
1.9647966114112223 3.6486819903861796
3.6486819903861796 8.070809886087057

For Cluster : 3
0.5446947326086805 -0.13350624247410142
-0.13350624247410142 2.0944731891517288

Purity is :0.764
NMI :0.07560366608611019
Cluster 0 size :247
Cluster 1 size :107
Cluster 2 size :146



For dataset3

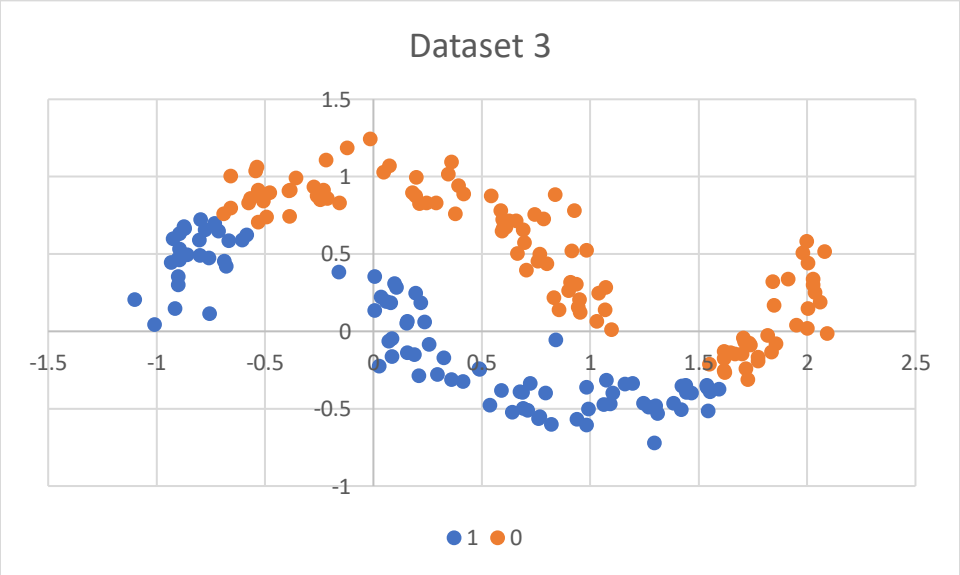
Number of Iterations = 96

After Calculations
Final mean =
0.7464039873937154 0.4563657579669118
0.28277046061942607 -0.05977930476972137

Final covariance =
For Cluster : 1
0.7693509663637754 -0.287832656314104
-0.287832656314104 0.19017543472044118

For Cluster : 2
0.6827225520222656 -0.30056294036671155
-0.30056294036671155 0.17581372396453213

Purity is :0.69
NMI :0.07594783950403936
Cluster 0 size :106
Cluster 1 size :94



The strengths of using Gaussian Mixture Models are that it is capable of handling clusters of varying densities and sizes. This particular characteristic wasn't particularly useful on the given datasets as all the ground-truth clusters were essentially the same density and size as each other. The most obvious weakness of GMM was the run time. The other two algorithms returned with almost imperceptible delay while GMM took almost a minute to return its results. This became especially noticeable with the second dataset which took several times longer than the other two datasets despite only containing twice as many examples indicating the algorithm's exponential time complexity. Two more weaknesses of GMM are than it has trouble estimating the number of clusters and is weak to non-spherical clusters. These are demonstrated by dataset 2, in which GMM split the cluster into 3 ,and dataset 3 in which GMM drew two ovals over the crescent clusters. A final weakness of GMM is that it only finds a local optimum and so is dependent on its starting conditions. It is not immediately apparent whether achieving the global optimums in the given datasets would have improved the final NMI but in general achieving the best results usually requires random restarts to improve the chance of reaching the global optimum.