Group:

Phillip Magnicheri

Souleymane Camara

Elisha Ngolemasango

PS3 report

**Methods:**

The methods used in PS3 included PCA, Kmeans, and the use of adjacency matrix. We used different variables inside both kmeans and PCA. As we went about testing different scenarios we used different random states, max\_iter(number of iterations), and n\_init(number of times centers were tested). For PCA we used random states, different numbers of components, and svd\_solver. For PCA we wanted to test and see if we reduced the number of dimensions to different values if it would change the accuracy. We started out randomizing PCA using random\_state but we found better results using svd\_solver to change the solver for PCA to arpack. Arpack is a package designed to compute a few eigenvalues and corresponding eigenvectors of a general n by n matrix. We used it since we were dealing with a decently large matrix and it seemed to take the randomness out of PCA which stabilized our results. Finally we used a adjacency matrix which was created by making a

2708 by 2708 matrix which corresponds with the data\_cites file for the cora documents given. We also used a standard scaler to better scale out starting data.

To test out results we used 4 different methods. The first was a simple LBL mapping which took a bincount of kmeans model based on a truth labels and then produced a matrix that showed the truth labels of our model. We then used the Hungarian algorithm to test our accuracy in our model compared. This algorithm compared the output of our model compared to the actual values for data given. We first took the bin count of our models labels and mapped them using labels given by the cara data set. We then used the linear sum assignment in the scipy package to get our mapping. We then ran the built in accuracy function in sklearn to gain the accuracy of our model. Lastly we used the built in function to gain the NMI (normalized mutual info score) and ARI(Adjusted Rand score). Finally to visualize our model we used matplot packers to make a scatter plot as well as seaborn to make a more clustered scatter plot.

**Start and improvements:**

We started out just using kmeans clustering all our data from df\_contents into 7 clusters. We started by using the no changes to the kmeans algorithm offered by sklearn. That includes leaving the number of iterations at 300 , the number of times the centers were tried at 10, and the letting kmeans choose a random start. At the beginning we settled on random state 99, which gave us:

**Starting Results:**

LBL Mapping:

[2, 4, 4, 0, 6, 1, 5]

Hungarian Accuracy:

0.43353028064992616 or 43.4%

Standard Metrics

NMI: 0.240

ARI: 0.144

To start improvements we first started by scaling the df\_nodes data we took from the cora contents file by using the standard scaler provided by sklearn. We then took that scaled data and ran it through our PCA function. Deciding on the \_ number of components through trial and error, that trial and error includes a test for components ranging from 2 all the way to 2708(the size of our matrix). We also tried using .99 as the components and changing our svd\_solver too full in the hope to better explain 99 percent of the data point. We then decided on arpack as the svd\_solver over the randomized class since it provided more stability.

After this we went one step further to make an adjacency matrix which was created by making a2708 by 2708 gathered from the connection gathered from cora sites file. This file was provided when a document was referenced and who referenced said document. After making this matrix we multiplied that matrix together with the matrix made by the PCA functions. This ended up giving us a 2708 by 15 which we then.

Finally to finish off our improvement we increased the n\_init(the number of times ran with center points) and max\_iter (number of iteration kmeans ran for one run) to make sure we were giving kmeans enough time to better determine the clusters as well as the center points. Again this was done by trial and error and we finally came to the number for n\_init being 50 and max\_iter being 500 since we saw only marginal increases going above these thresholds. The results when using these 2 variable resulted in:

**Final Results**

LBL Mapping:

[0, 1, 6, 3, 4, 5, 6]

Hungarian Mapping:

[0 1 2 3 4 5 6]

Hungarian Accuracy:

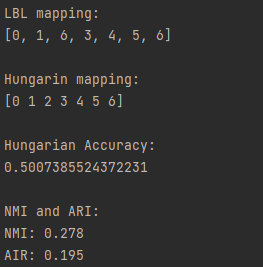
0.5007385524372231 or 50.1%

Standard Metrics

NMI: 0.278

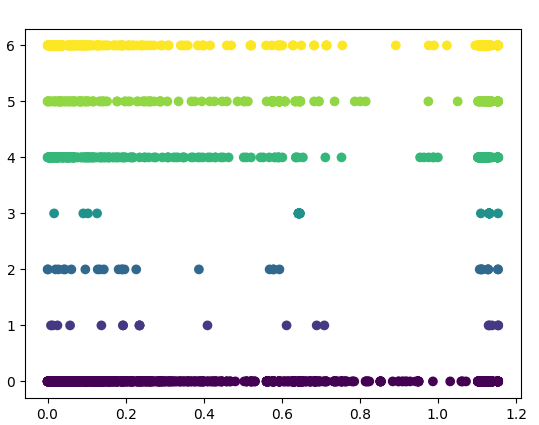
AIR: 0.195

**Console results:**

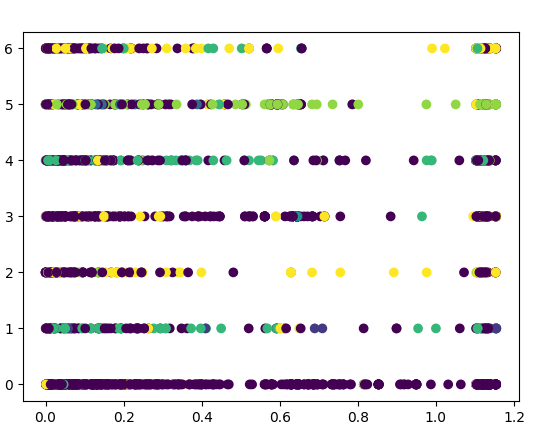
****

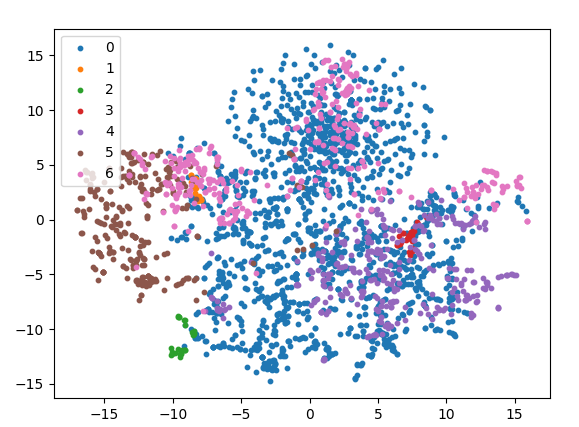
**Visuals:**

**Scatter plot(groups all points by model label):**

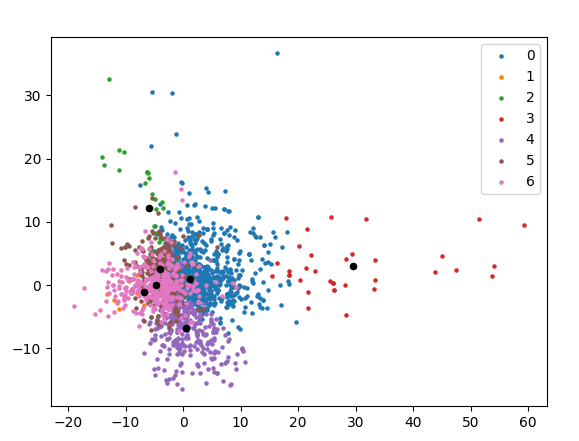


**Scatter plot(groups all points by actual label with color of model):**



**Scatter plot of clusters colored by grouping using t-sne and seaborn:** 

**Scatter plot of clusters colored by grouping w/clusters:**



**Limitations and future improvements:**

With the model we are presenting in this report the limitations are the accuracy at which specific points can be calculated. As the results show for the hungarian mapping says we have found all 7 unique labels and in the correct order but there are 2 things that server;ly limit these results. The first being that when you look at the LBL mapping you can see that there is some confusion when it comes to identifying label 2 and 6. The second being the visuales that have been presented. Our model has confidence when it comes to find labels: 0,4,5,6 but has very limited accuracy finding 1,2,3. This tells me that our model mis-classifies labels that should be 1,2,3 and that can hurt future results when more data is added. There are many avenues that could improve our model. A start would be to use the given adjacency matrix to calculate a page rank and find better cluster centers. The next step would be to implement a graph auto encoder, this would hopefully improve and define each data entry. That would hopefully distinguish classes 1,2,3 better from the other classes and give over all better clusters. We want to reduce the dimension of our data dn get a better representation of it. The hope would be that the auto encoder would better latent vectors and then we could reduce our vector better without losing meaning.