Willy Esquivel-lopez – G01127937

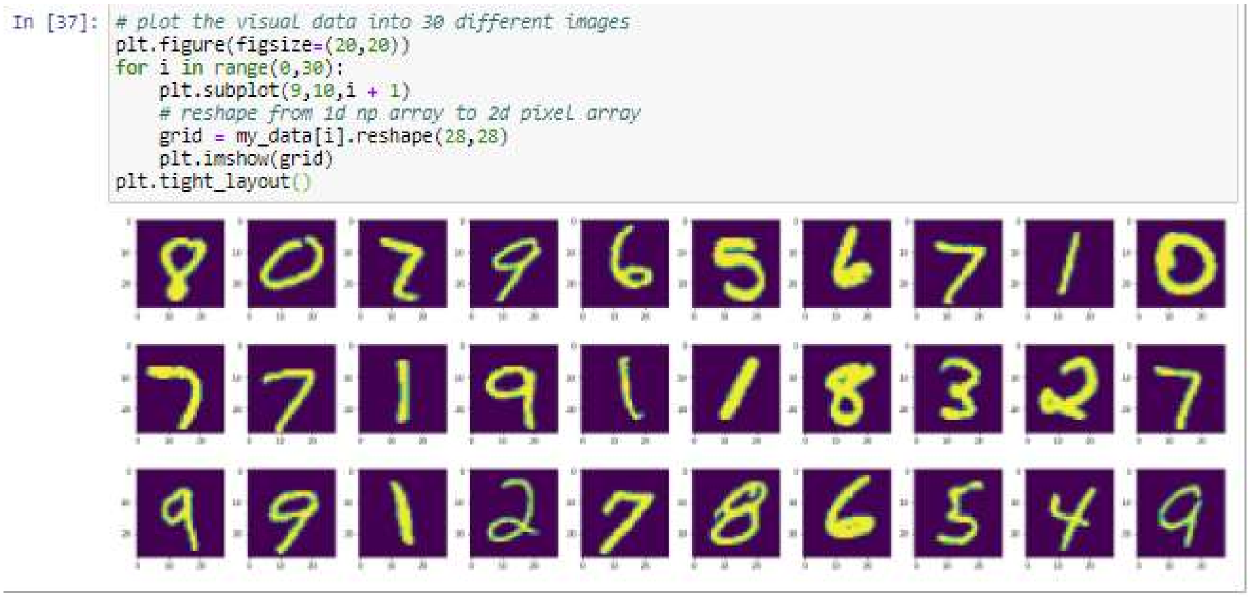
Miner2 username: Gopher123

Best Public Score: Phase 1 = .80, Phase 2 = .79

HW4 Report

My approach:

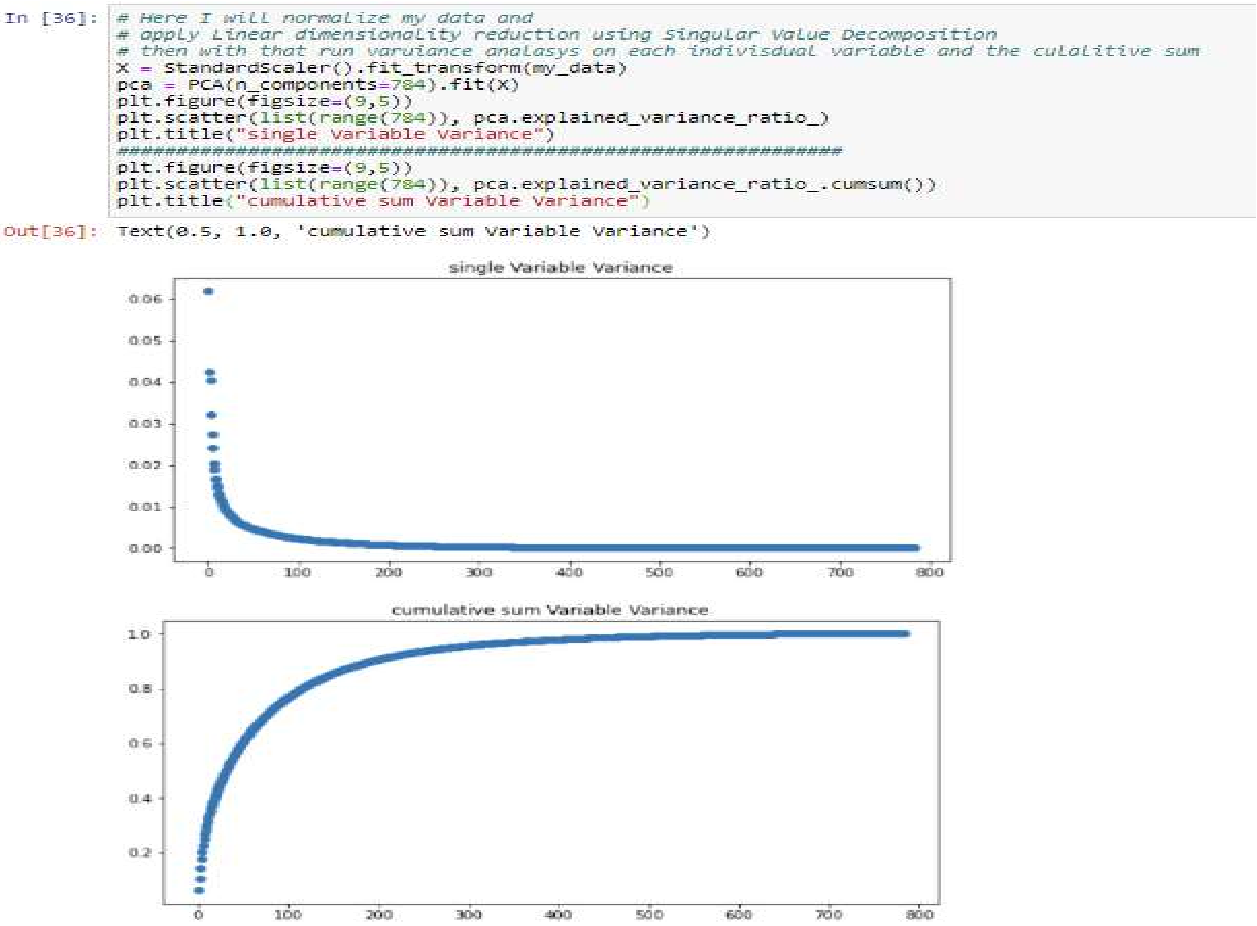
When seeing that the small dataset for Iris and the Visual Data set for Phase 2 was unlabeled and that the target value was not given, I could no longer use the traditional train test split among x and y axis. upon closer examination of the data, I could still not find any correlation at first glance for either of them. I plotted the visual data using a 28x28 grid to visually understand the data and any trends.



```
In [37]: # plot the visual data into 30 different images
         plt.figure(figsize=(20,20))
         for i in range(0,30):
             plt.subplot(9,10,i + 1)
             # reshape from 1d np array to 2d pixel array
             grid = my_data[i].reshape(28,28)
             plt.imshow(grid)
         plt.tight_layout()
```

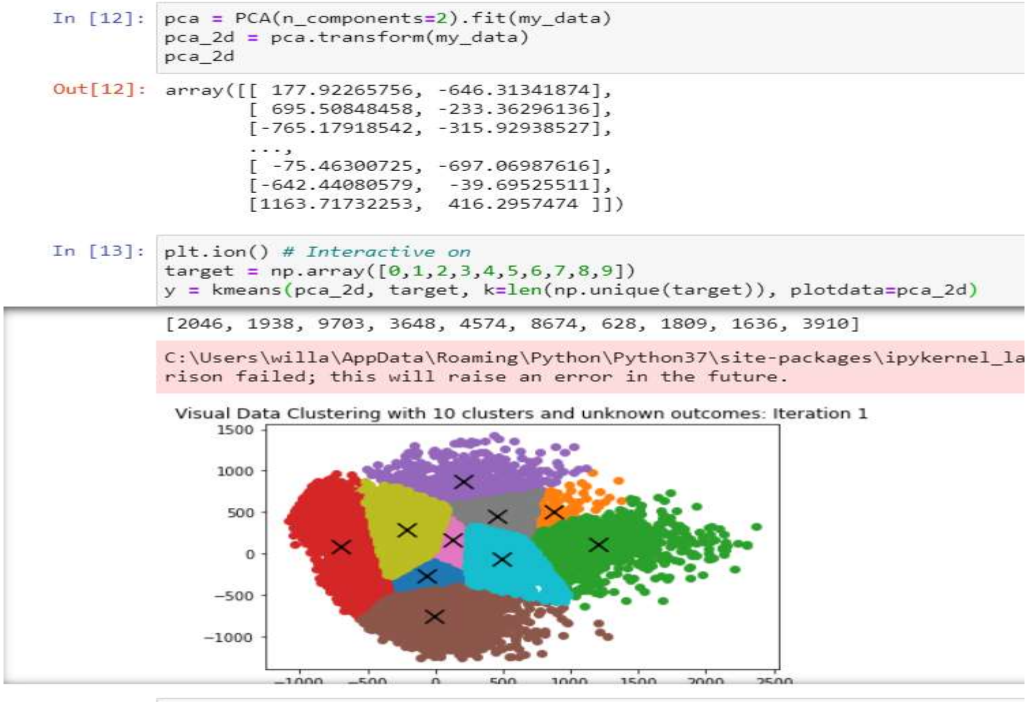
Analysis:

I tried plotting it to make sense and began estimating some numbers such as mean, std div, and generating some random centroids given those values. But given all that information I could not generate any useful correlation or knowledge. Thus, I looked for other means of plotting the data, I began trying to solve for the variance and discovered a library “sklearn.decomposition.PCA” which I could use to reduce the dimensionality of my data by using Singular Value Decomposition. The Lower Dimensional space, which would now allow me to analyze variance on single or the cumulative sum of my variables by fine tuning the number of components I can use.



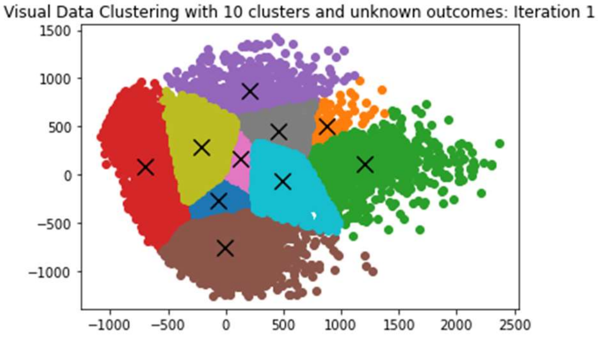
```
In [36]: # Here I will normalize my data and
         # apply Linear dimensionality reduction using Singular Value Decomposition
         # then with that run varuiance analasys on each indivisdual variable and the culalitive sum
         X = StandardScaler().fit_transform(my_data)
         pca = PCA(n_components=784).fit(X)
         plt.figure(figsize=(9,5))
         plt.scatter(list(range(784)), pca.explained_variance_ratio_)
         plt.title("single Variable Variance")
         #############################################################
         plt.figure(figsize=(9,5))
         plt.scatter(list(range(784)), pca.explained_variance_ratio_.cumsum())
         plt.title("cumulative sum Variable Variance")

Out[36]: Text(0.5, 1.0, 'cumulative sum Variable Variance')
```

This was especially useful as it gave me a better insight as to what the variance was for each component and could determine the estimation of number of clusters in my data and each component variation. Now with this given I began to Implement My K-Means algorithm in a fashion where I would be able to see the plotting of each iteration and the K-Means progression in clustering the data. Thus, I applied PCA and ran my K-Means algorithm.

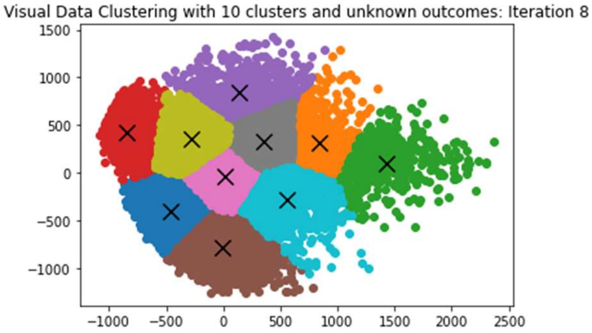


```
In [12]: pca = PCA(n_components=2).fit(my_data)
         pca_2d = pca.transform(my_data)
         pca_2d

Out[12]: array([[ 177.92265756, -646.31341874],
                [ 695.50848458, -233.36296136],
                [-765.17918542, -315.92938527],
                ...,
                [ -75.46300725, -697.06987616],
                [-642.44080579,  -39.69525511],
                [1163.71732253,  416.2957474 ]])

In [13]: plt.ion() # Interactive on
         target = np.array([0,1,2,3,4,5,6,7,8,9])
         y = kmeans(pca_2d, target, k=len(np.unique(target)), plotdata=pca_2d)

         [2046, 1938, 9703, 3648, 4574, 8674, 628, 1809, 1636, 3910]

         C:\Users\willa\AppData\Roaming\Python\Python37\site-packages\ipykernel_la
         rison failed; this will raise an error in the future.
```

Visual Data Clustering with 10 clusters and unknown outcomes: Iteration 1
1500
1000
500
0
−500
−1000
−1000
−500
0
500
1000
1500
2000
2500


Visual Data Clustering with 10 clusters and unknown outcomes: Iteration 1

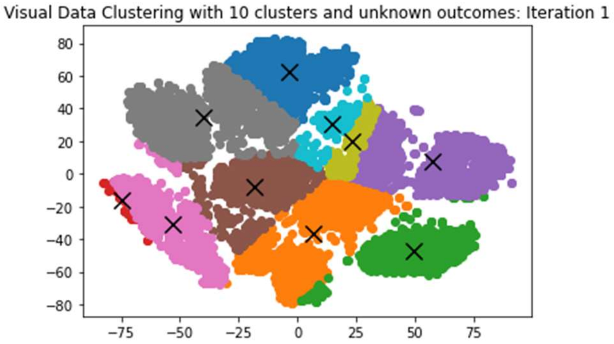
->

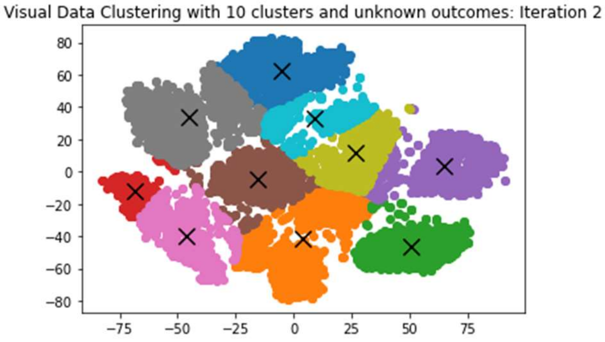

Visual Data Clustering with 10 clusters and unknown outcomes: Iteration 8

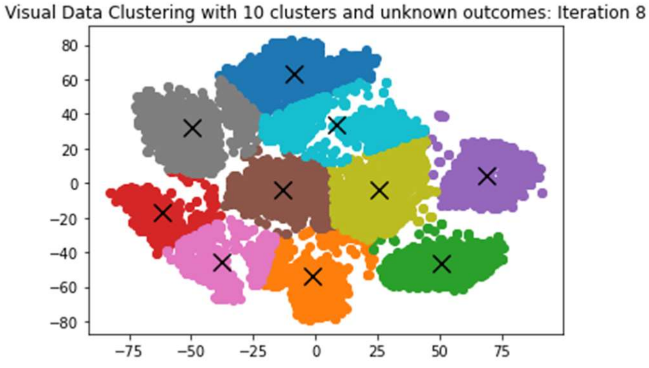
Directly applying the dimensionally reduced data was not making any big changes in term of iterations, as the data points were densely packed into one location. This was still much better than not using Singular Value Decomposition, but I knew that this would be improved.

Applying and testing K-Means with T-SNE:

Unfortionately the best scores I could get with Singular Value Decomposition preprocessing was around mid to low 60%. So I had to think of another way to cluster my datapoints. I began to do reaserch on other means of representing my clustering of my data. Because without using PCA for Singular Value Decomposition, I would average scores below 50%. I the found t-distributed Stochastic Neighbor Embedding as an alternative to reduce high dimensional data. It uses similarities between datapoints to produce joint probabilities, making it effective on high-dimensional visual data. I used this for preprocessing the data before applying my K-Means algorithm and averaged a V-score of ~70% on my prediction, making it the highest so far. I continued to iterating through the cluster centroids outcomes, seeing if I could find a way to improve the measurement in terms of randomly initializing good centroids. The best guess so far was initialized at centroid locations: [5385, 6835, 100, 3269, 9512, 7545, 3119, 586, 7236, 866] K-value 6 with the following classes plotted:

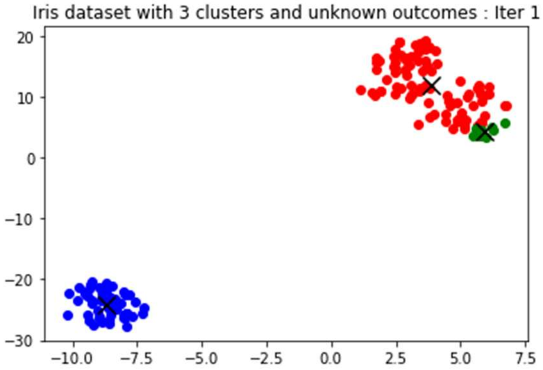

Visual Data Clustering with 10 clusters and unknown outcomes: Iteration 1

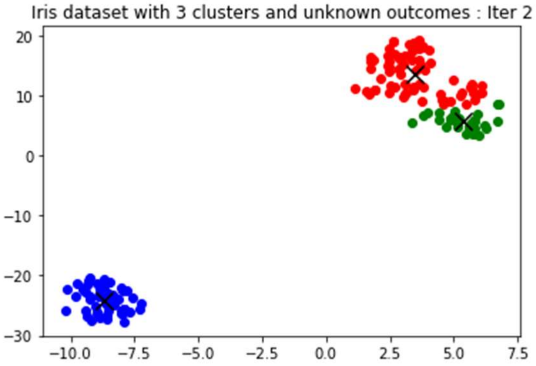

Visual Data Clustering with 10 clusters and unknown outcomes: Iteration 2

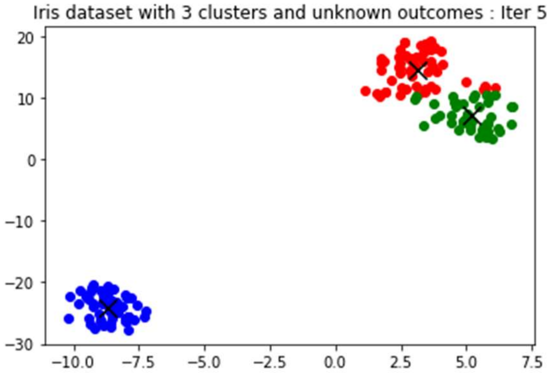

Visual Data Clustering with 10 clusters and unknown outcomes: Iteration 8

This was a result of using K-Means in 8 iterations, as 8 iterations is the point in which the algorithm performs best. Resulting in a V-score of 76% for the image dataset

Using the two approaches on the Iris data set had similar but far better scores given the more well-defined clusters resulting in a best V-Score of 80%.







Conclusion:

In terms of the K means algorithm I found that not using any form of reduction was not an option, resulting in V-scores of ~30%, using PCA for Singular Value Decomposition preprocessing resulted in V-scores of ~60%, and finally applying T-SNE resulted in V-scores of ~70%. When using both in combination Iris dataset did not make much progress staying upper ~70% regardless the number of components that were used in the Singular Value Decomposition, as I could not tune the number for T-SNE with my implementation of K-Means. Although the pre-processing took nearly three times longer using T-SNE as opposed to PCA. Finally, my K-Means agorithm proved to work effectively with T-SNE and PCA in combination on the Image Dataset resulting in a score of 79% when using two froms of dimensianality reduction I was required to increase the number of iterations for the K-Means algorithm keeping K = 10 and number of iterations at 16, as the clusters were initially more defined and their centroids were more alligned with the respective targets.

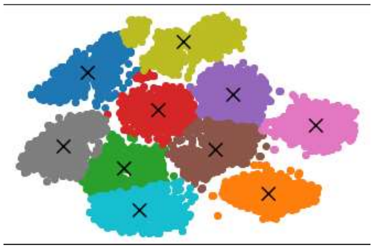


Figure 1- Best Result of clustering visual data.