DIGIT RECOGNITION PROJECT

Emre Can Şen-21902516

Question-1

For this question and all the other questions as well sklearn module was used since it is the go to data science and machine learning library, so sklearn will be heavily utilized through all questions. Additionally, matplotlib library was used for the plots. The data was split to two as train and test via train_test_split function. Then sklearn's PCA, Gaussian functions were utilized to complete the respective steps of the question.

1.1- The instructions were followed, and the mean of the whole data from each sample was subtracted.

1.2- Then PCA is applied with sklearn library and eigenvalues are plotted.

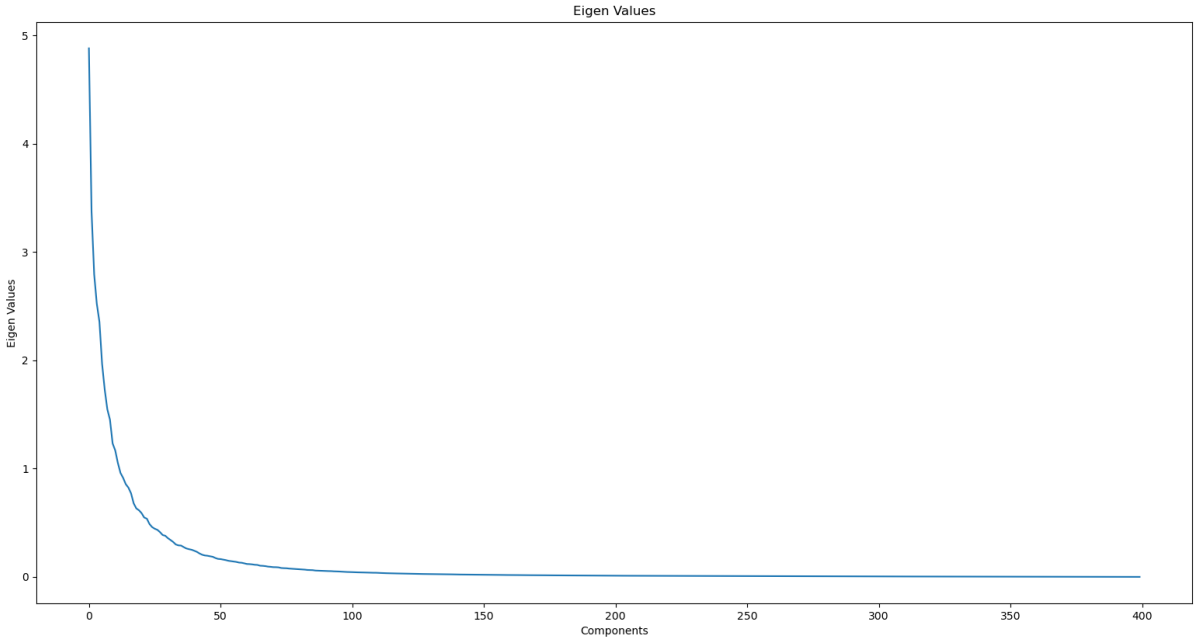

Figure 1- Eigenvalues vs. Components

Looking at the plot, it can be seen that most of the eigenvalues are included up to 60-70 component range. After this range, the rate of inclusion continues to drop significantly. So the 70 components point seems enough for the purposes of the question.

1.3- The components size was 400 so the image was reshaped to 20x20 array and then transposed to get the image:

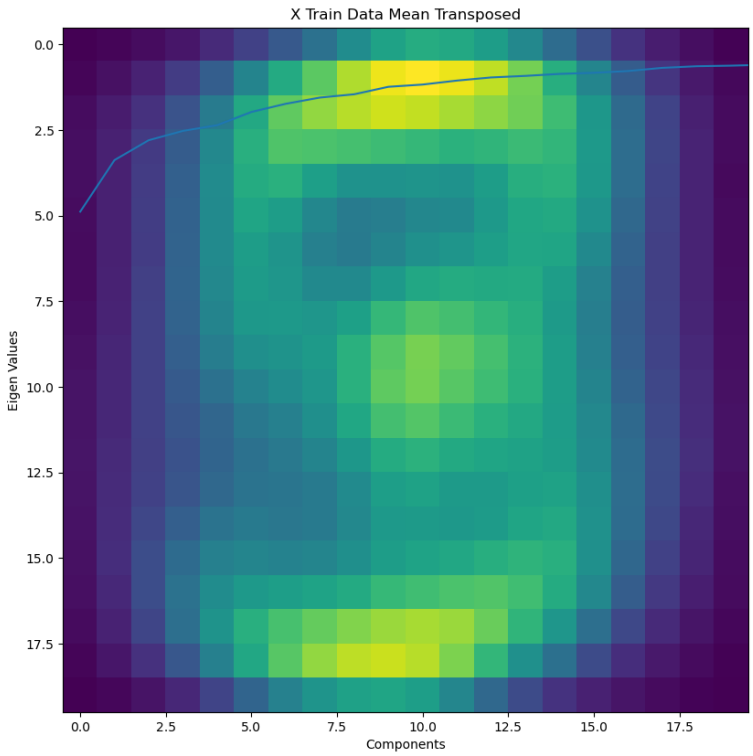

Figure 2- Mean Image of the Data

The image looks like the combination of numbers 3 and 9. Also from the image the number 8 and 5 can be sort of seen. So it can be said that most of the numbers consist of soft and curved lines, rather than sharp lines, since the properties of the mean image is curved and soft. Also the heat map shows that the middle top and middle bottom are the most used common areas when writing these numbers.

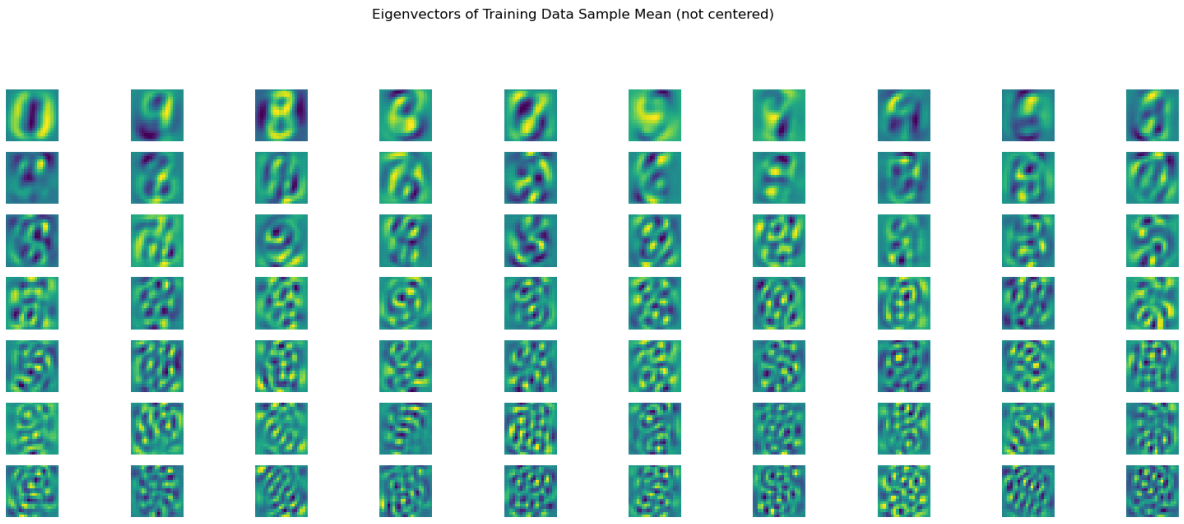

Figure 3- Base Images

Then the chosen 70 first principal components are drawn. Most of the images here also feature curved and soft lines and some of the numbers can be clearly seen in the initial images, the numbers 0, 9, 8 respectively etc.

1.4- Dimensions starting from 1 and going all the way to 200 are chosen and both the training and test data are projected to train a Gaussian Classifier.

1.5-

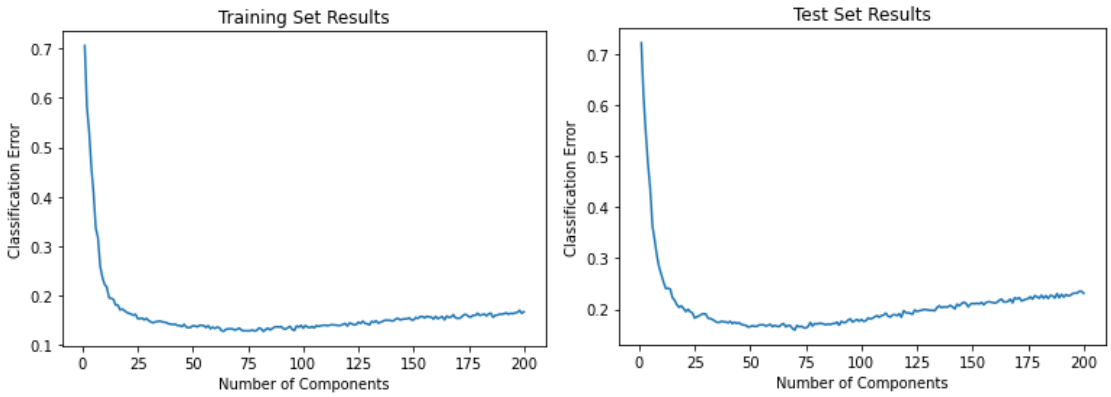

Figure 4- Training vs. Test Error Results

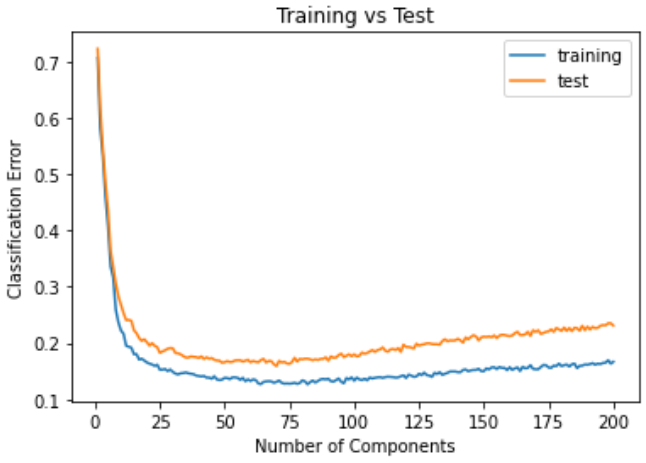

Figure 5- Training vs. Test Error Results Together

As it can be inferred from the figure 5, the test error is always higher than the training error. This outcome is as expected because the model included the training data when the model was trained. So it

should perform better when this data is again used for accuracy, unlike the test data which is more foreign.

Question-2

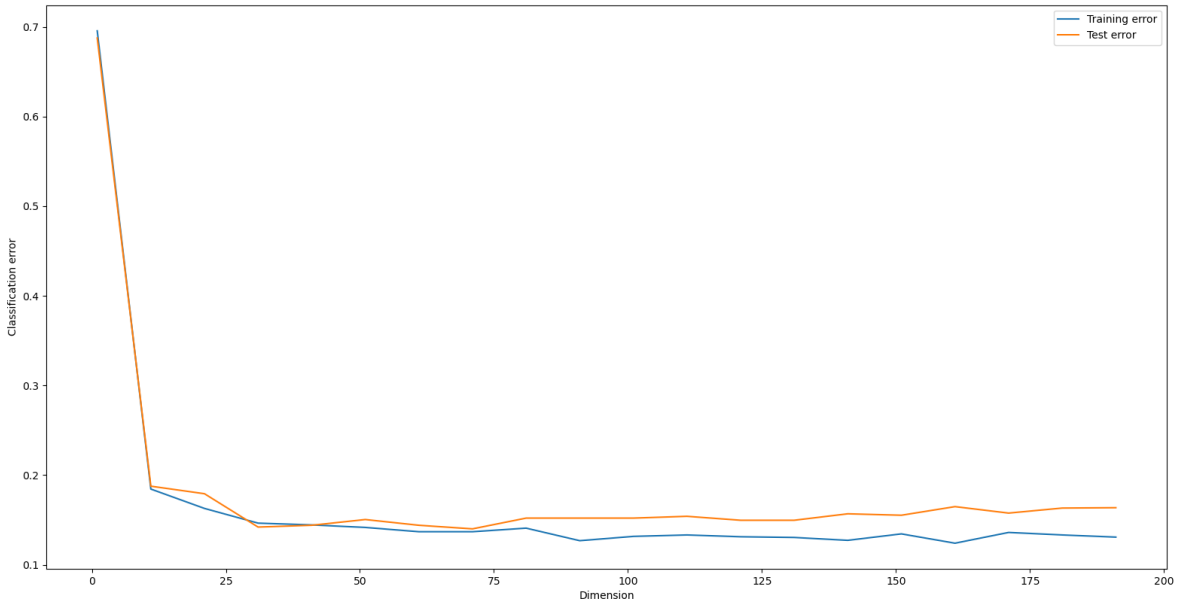

Figure 6- Classification Error vs. Number of Dimensions

Same with the PCA case, after some certain point more dimensions added does not mean better results. In fact, after some point the model starts to over-fit and error increases, this can be observed around 150 dimensions. Eigen_solver parameter was set to auto to choose the most efficient solvers. The lowest PCA error was 0.23, and lowest Isomap error was 0.14. So the Isomap outcome errors are lower.

Question-3

t-SNE is an unsupervised learning method and it is commonly used for visualizing high dimensional data. It is similar to PCA but t-SNE is nonlinear and its one of the best dimensionality reduction methods. To get better results, number of iterations and perplexity variables are experimented with:

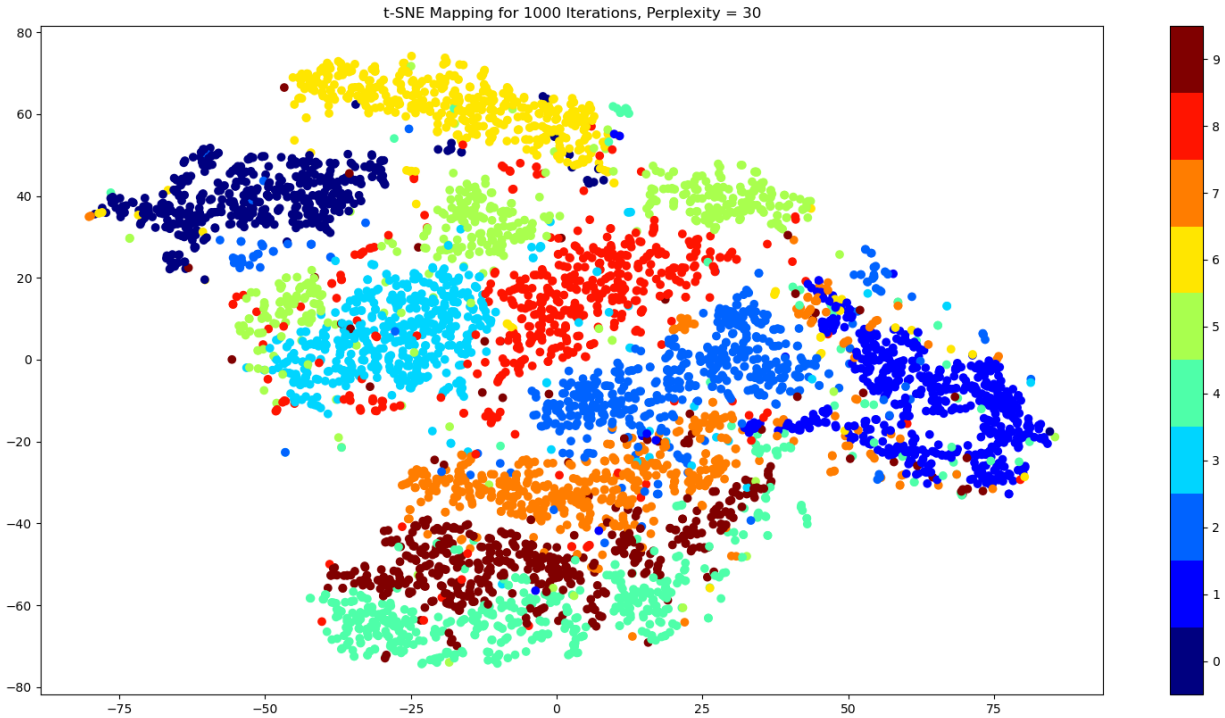

Figure 7- 1000 Iterations and 30 Perplexity t-SNE

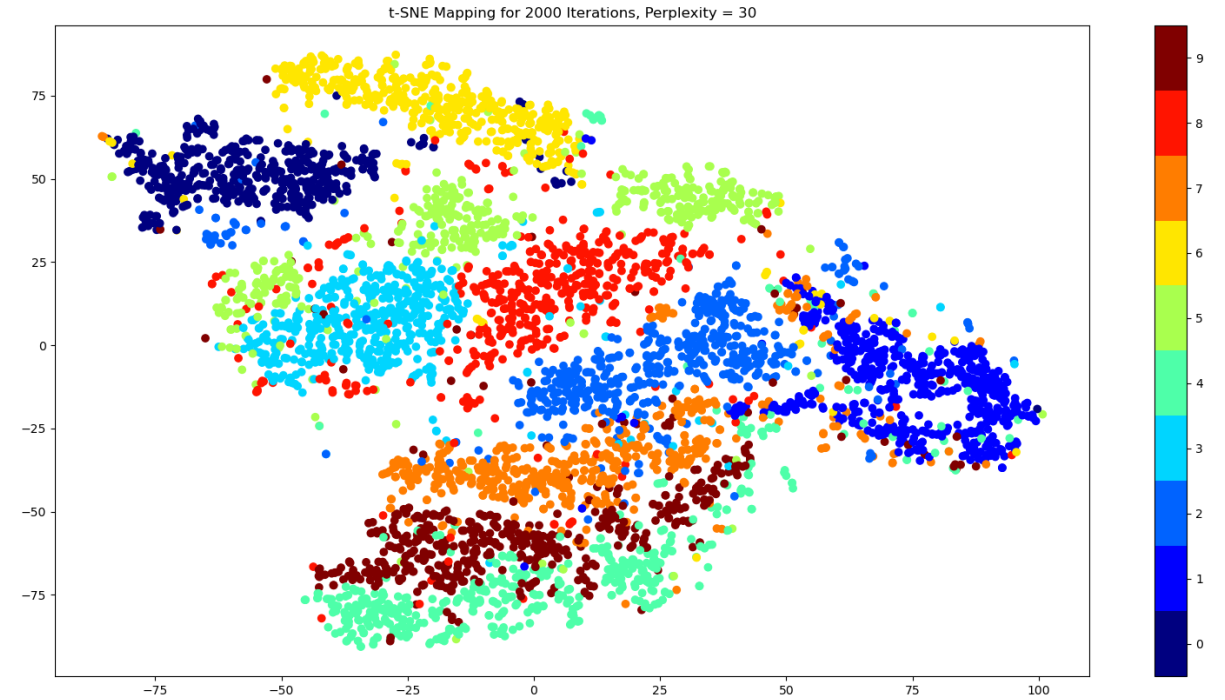

Figure 8- 2000 Iterations and 30 Perplexity t-SNE

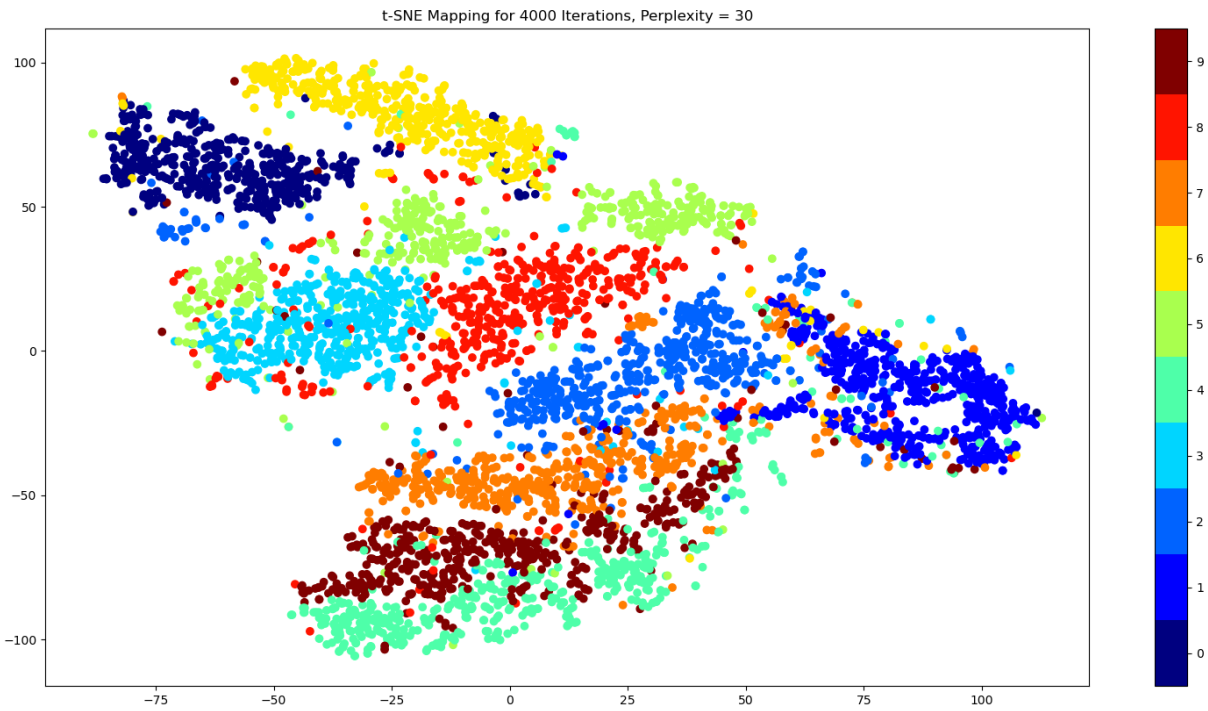

Figure 9- 4000 Iterations and 30 Perplexity t-SNE

So with more iterations, the clusters became more separated from others. This is most evident in the dark blue (0) class. The data points get closer to same color points, and get away from other colored points. However, some other data points, like the light blue (3), don't get far away from the other classes. All in all the effects seem to affect different classes with differing amounts and in ways.

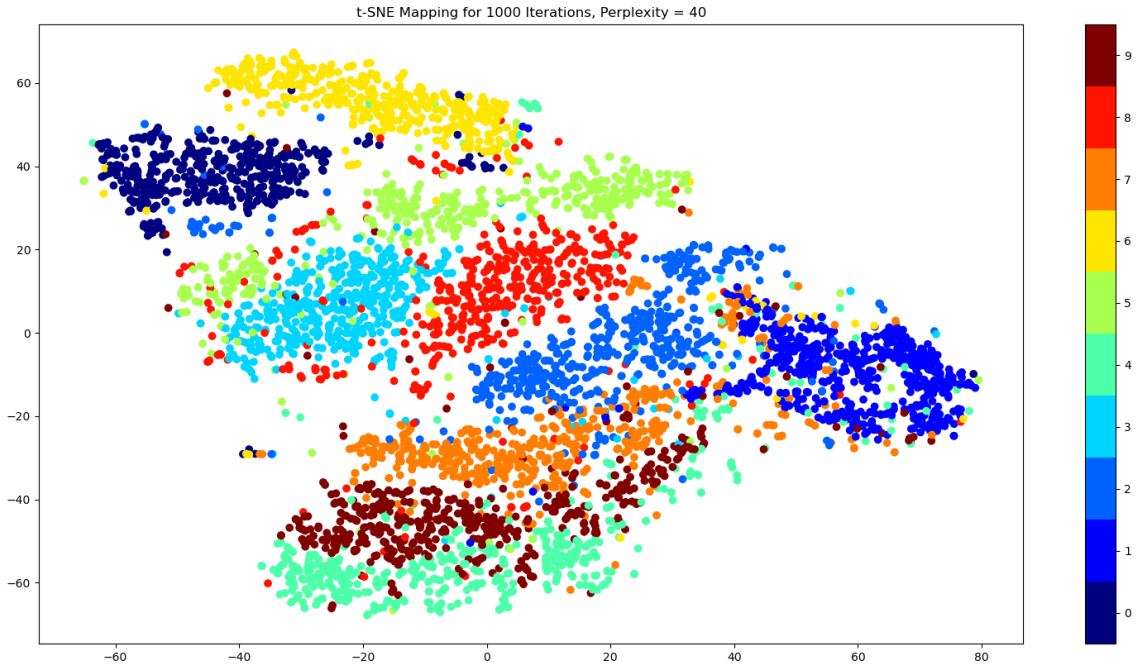

Figure 10- 1000 Iterations and 40 Perplexity t-SNE

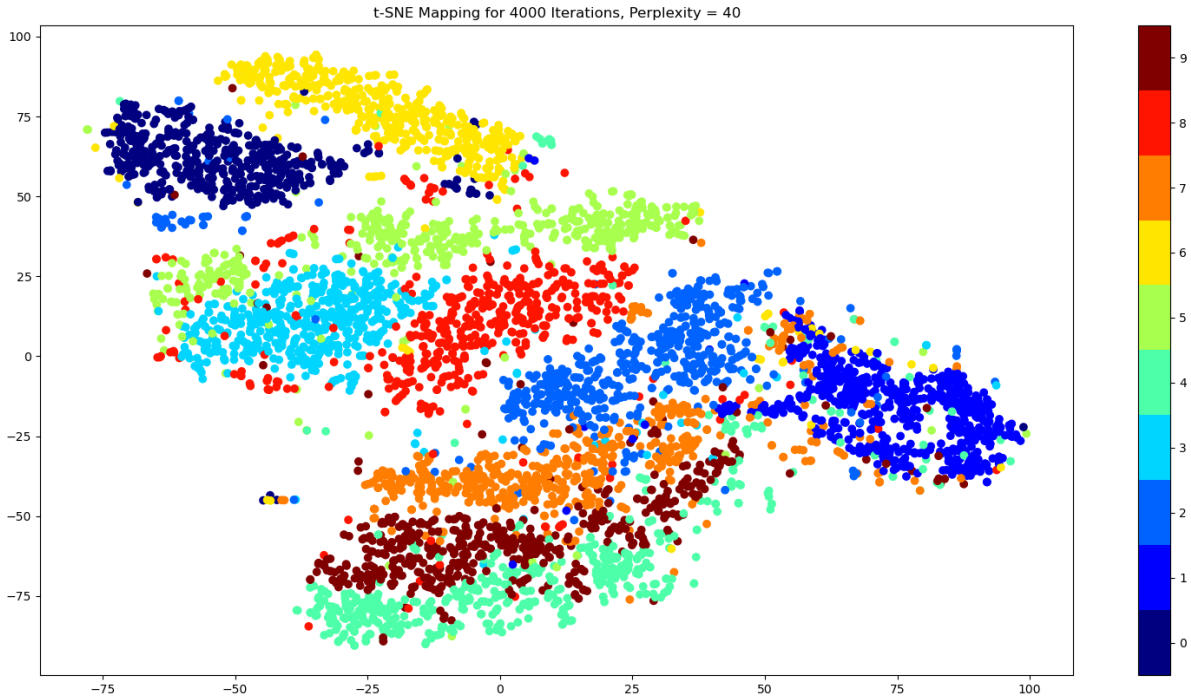

Figure 11- 1000 Iterations and 40 Perplexity t-SNE

Perplexity parameter controls the number of nearest neighbors. When it increases that data points compress. From the perplexity experiments, and in the figures 10 and 11 this fact can be observed.

References

Numpy

Sklearn : https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html