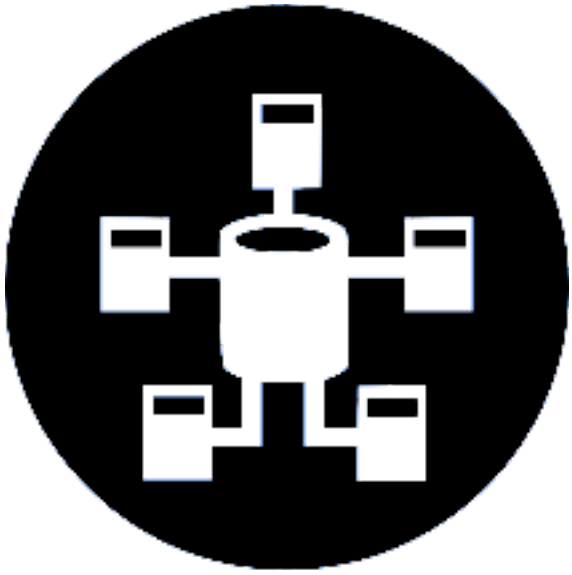


Clustering Phase

(Assignment-4)

Student1: Sarthak Singhal(20171091)

Student2: Sajal Asati(20171183)

Data set used: MNIST

Part 1) Dataset and the cluster output

- We have used the MNIST database of handwritten digits, which is a subset of a larger set available from NIST. The digit images have been size-normalized and centered in a fixed-size image. We took a sample of 6000 data points with 784 dimensions (28x28 images).
- The source of the exact dataset file is our SMAI course assignments and homework problem sets.
- **Final plot approach:** The dimensions of the original dataset were reduced by using TSNE approach and then clustering was performed using **DBSCAN** algorithm.
- As it is a dataset consisting of all the digits, hence the final output clusters must contain 10 clusters and we are also able to get 10 clusters with few other small groups of points(due to closeness among clusters).
- Following Figure 4 is the output produced finally yielding all the clusters where circles represent dataset points which are part of the cluster and the ‘x’ represent the outlier points.

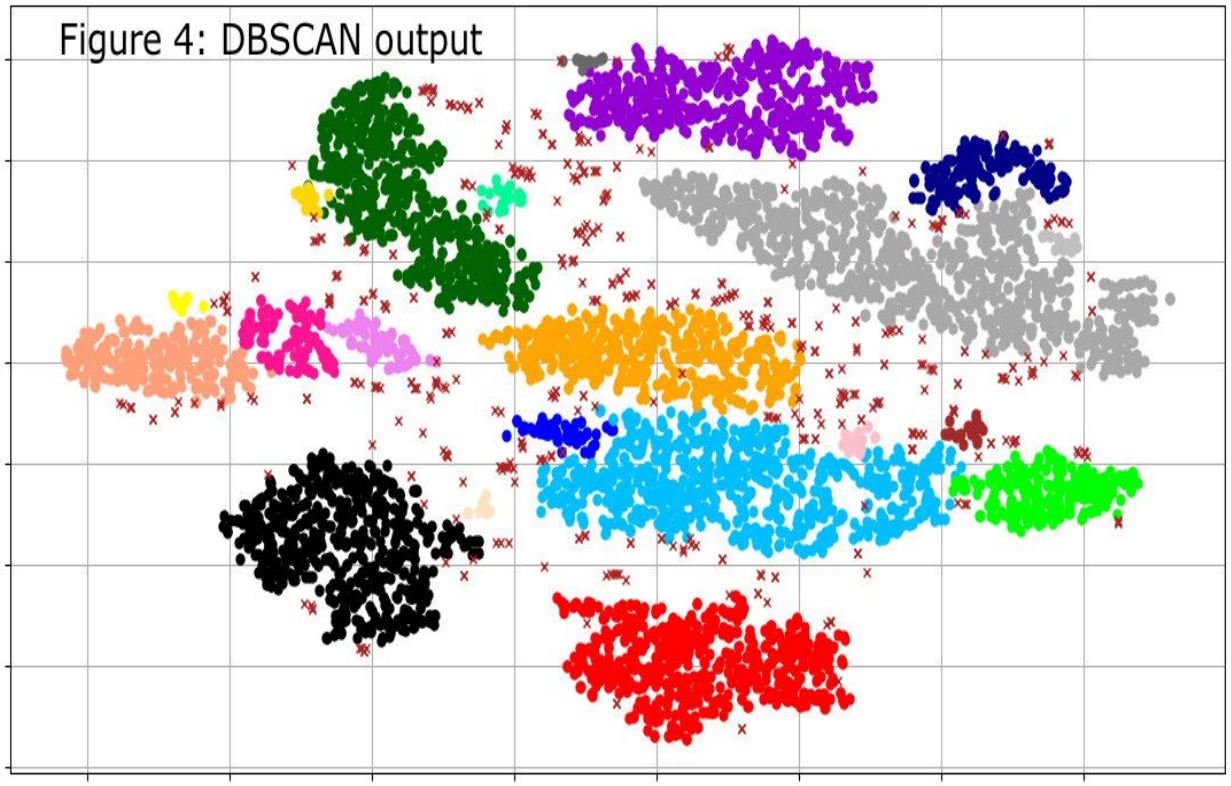


Figure 4: DBSCAN output

Part 2) Experiments done on dataset

- Following is the procedure described how we finally arrived at the final cluster output as shown in the previous part.
- First we reduced the dimensions of our dataset so that the algorithm can run in sufficient amount of time.
- As the data needs to be visualized in 2D(for convenience), and the original dataset contains 784 dimensions, hence first we used PCA for dimensionality reduction and the projected the data into the two most important Principal axes.
- Following Figure 1 is a plot of the original data in 2D(without class labels)

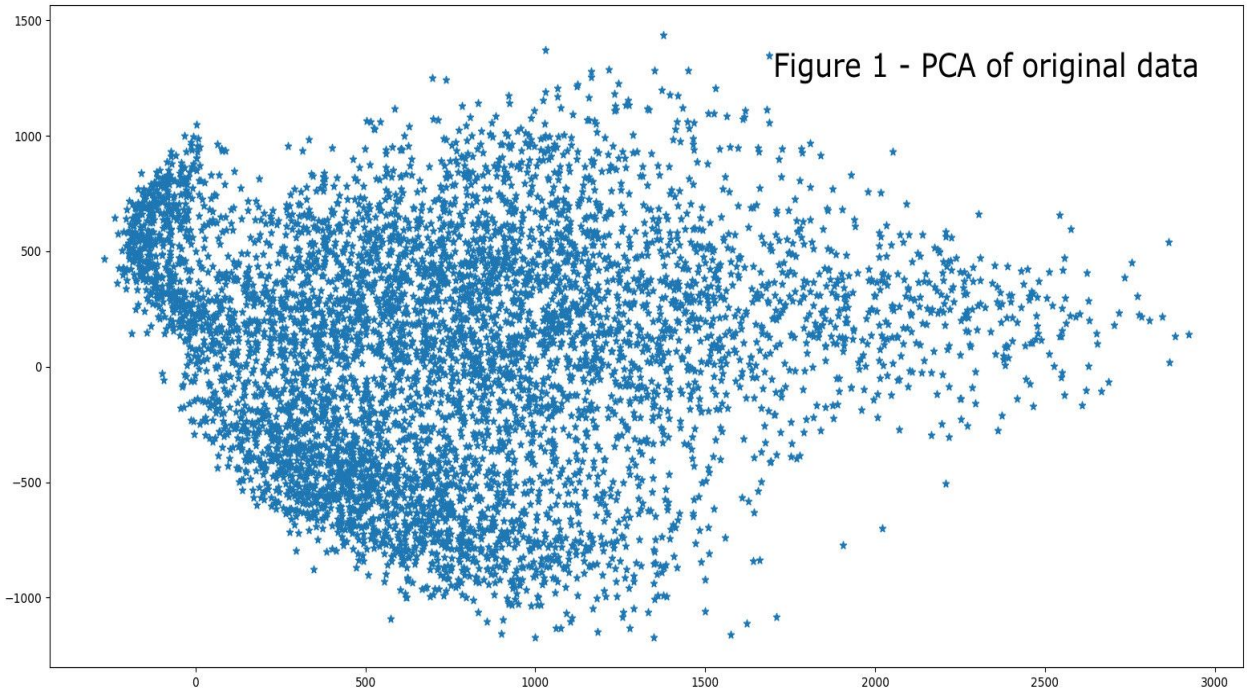
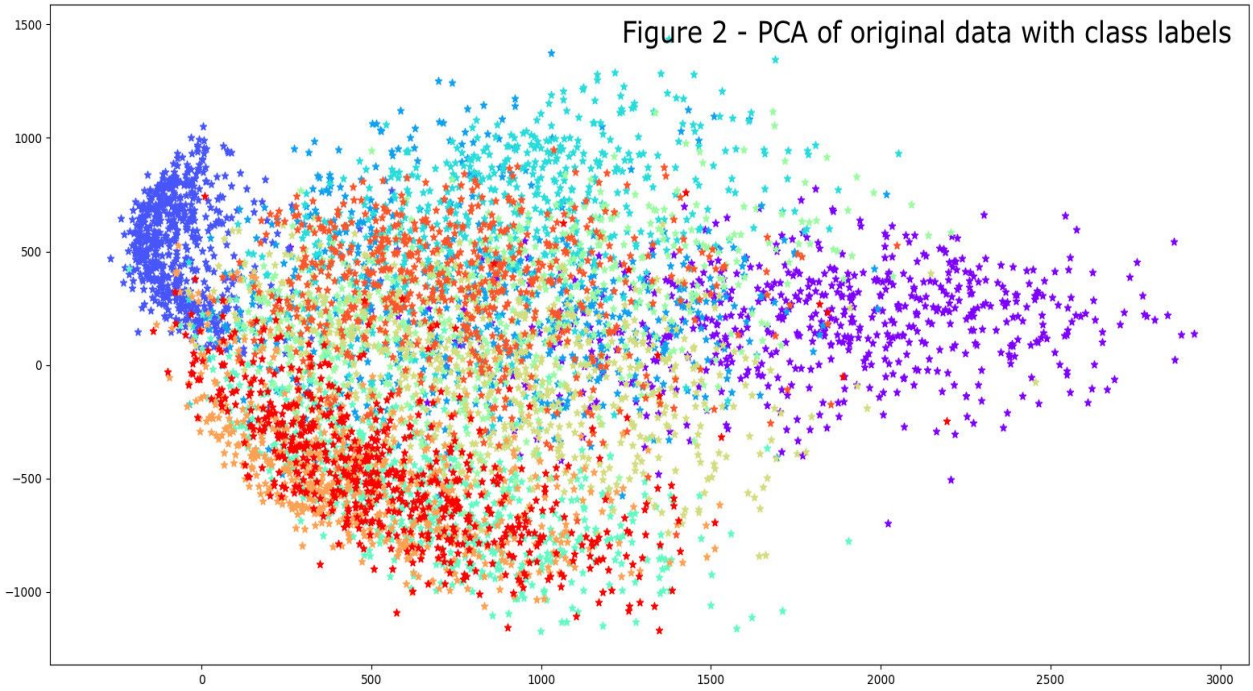
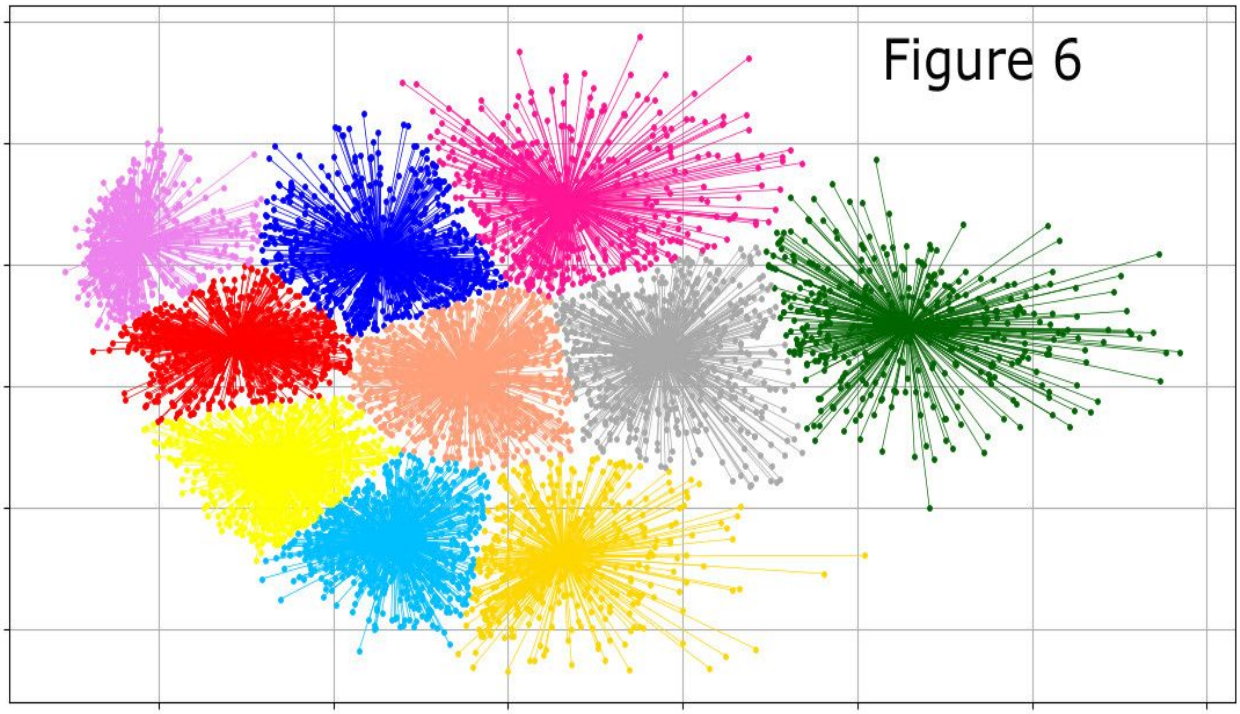


Figure 1 - PCA of original data

- And the next Figure 2 is the plot with class labels assigned to them - we took the classes with labels for clustering so that we can better analyze the results of PCA:



- As we can see that due to projection onto the two principal axes, there is significant overlapping of the samples of different classes. Hence any algorithm which tries to cluster this data will ultimately won't be able to give us clear cluster groups because of this overlapping.
- But we still tried applying k-means to the above reduced dataset (Figure 1) and we got the following output clusters: Figure 6 plot



- But on comparing this Figure 6 with Figure 2 plot it is clear that this k-means cluster output is nowhere close to what the original dataset had (in Figure 2).
- So now to reduce the dimension of data from 784 to 2 dimensions (which we surely need to do for the purpose of calculation), we then used **TSNE** (t-distributed Stochastic Neighbor Embedding)
- Using TSNE algorithm following are the plots of original data projected into 2 dimensions with and without individual class coloring:

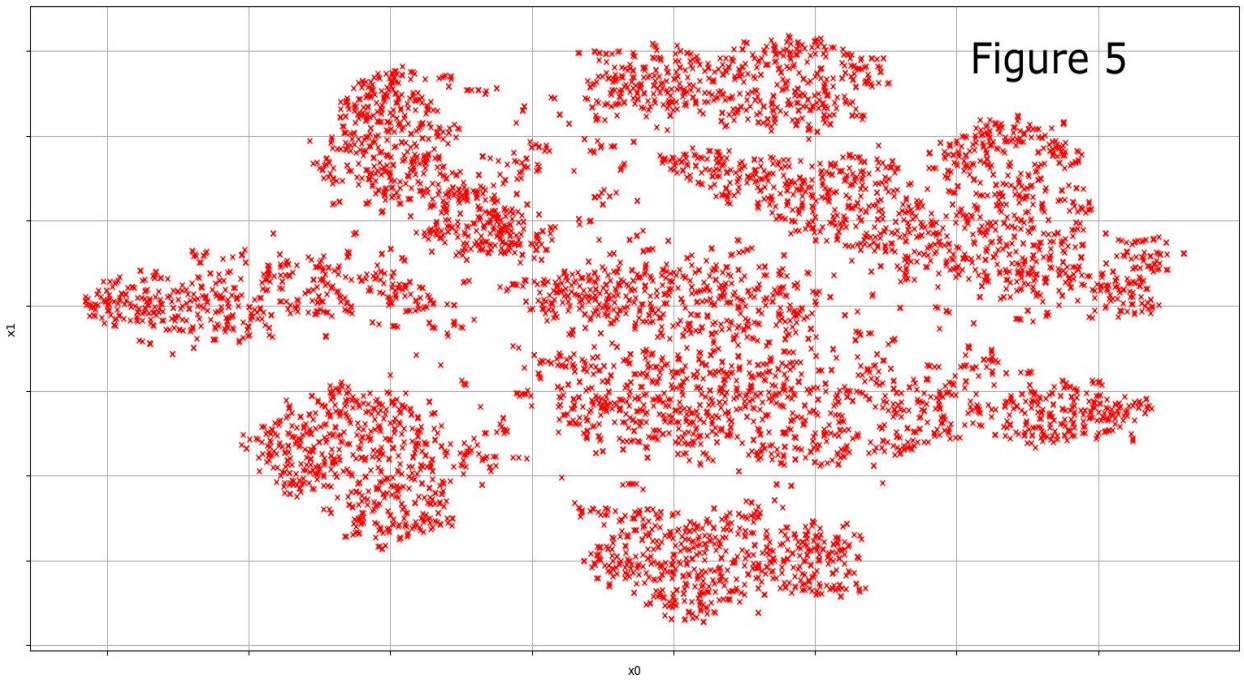


Figure 5
x1
x0

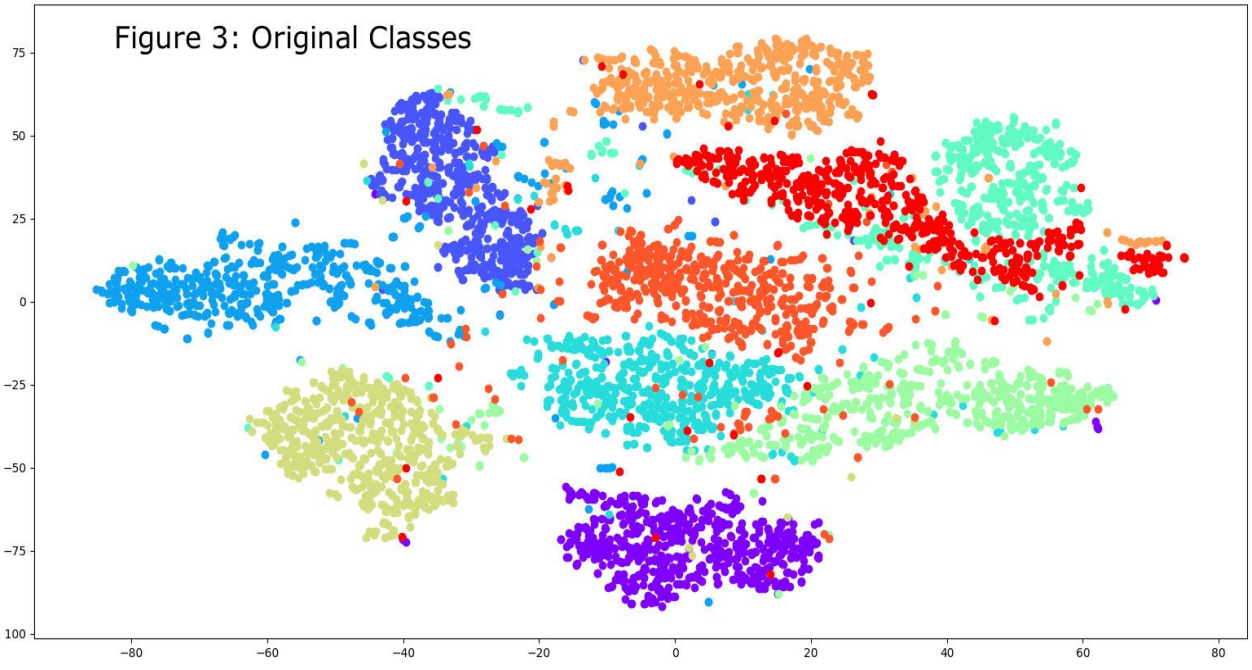
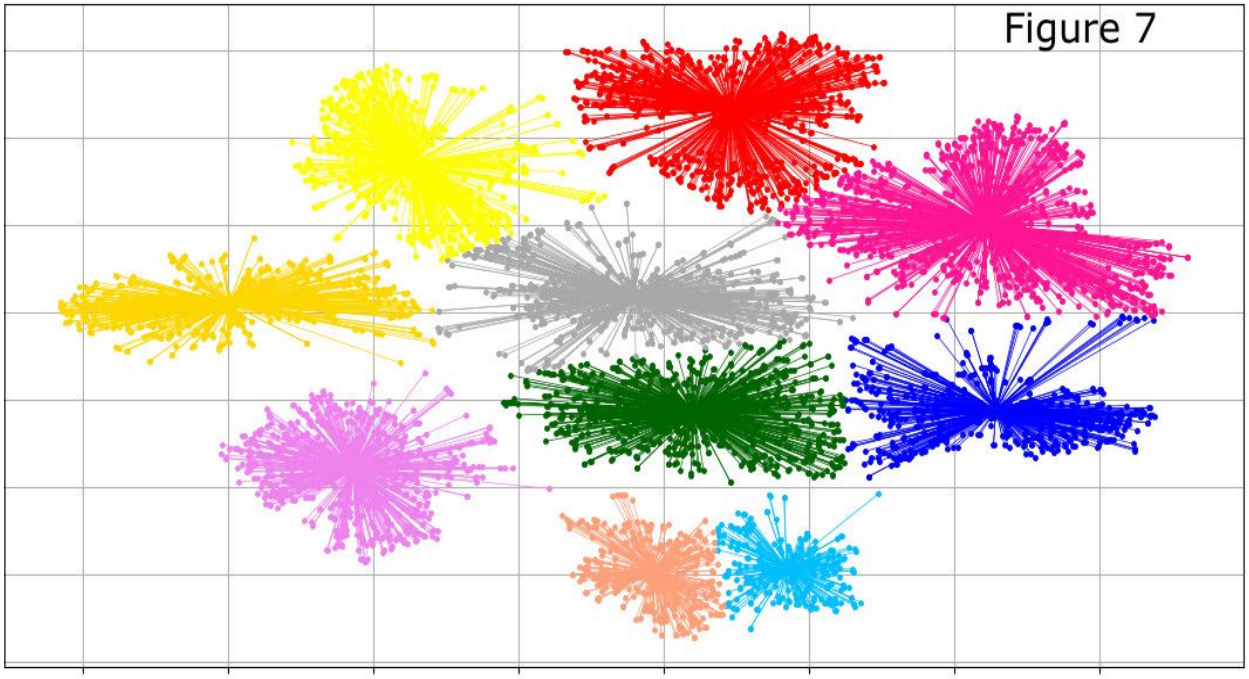


Figure 3: Original Classes
75
50
25
0
−25
−50
−75
100
−80
−60
−40
−20
0
20
40
60
80

- We can clearly see that this clusters the data much better than PCA approach. Though there is still some overlapping but the classes are now much more clearly separated than the earlier case.
- And hence, now applying clustering algorithms to this dataset should give us reasonable results.
- Keeping in mind that the data points are digits, the first algorithm we used was k-means with number of clusters = 10
- Following is the output of applying k-means:



- As the k - means tends to give us spherical shaped clusters (because of the euclidean distance property), hence we are getting slightly circular shaped clusters.

- Also k-means algorithm is unable to differentiate the clusters which are close to each other as it merges two top-right clusters into one big cluster.
- Also we got similar results using k-medoids algorithm.
- Next algorithm we tried (in the hope of finding a better cluster distribution) was DBSCAN. As it is a density based clustering algorithm, hence it might take advantage of the closeness of the points, and would be able to find clusters of any shape.
- We ran DBSCAN algorithm many times with different parameters to get a perfect cluster distribution. Finally the values of epsilon and number of points at which we were getting the best results were **2.93 and 11 respectively**.
- Following is the output plot of the cluster distribution from DBSCAN algorithm.

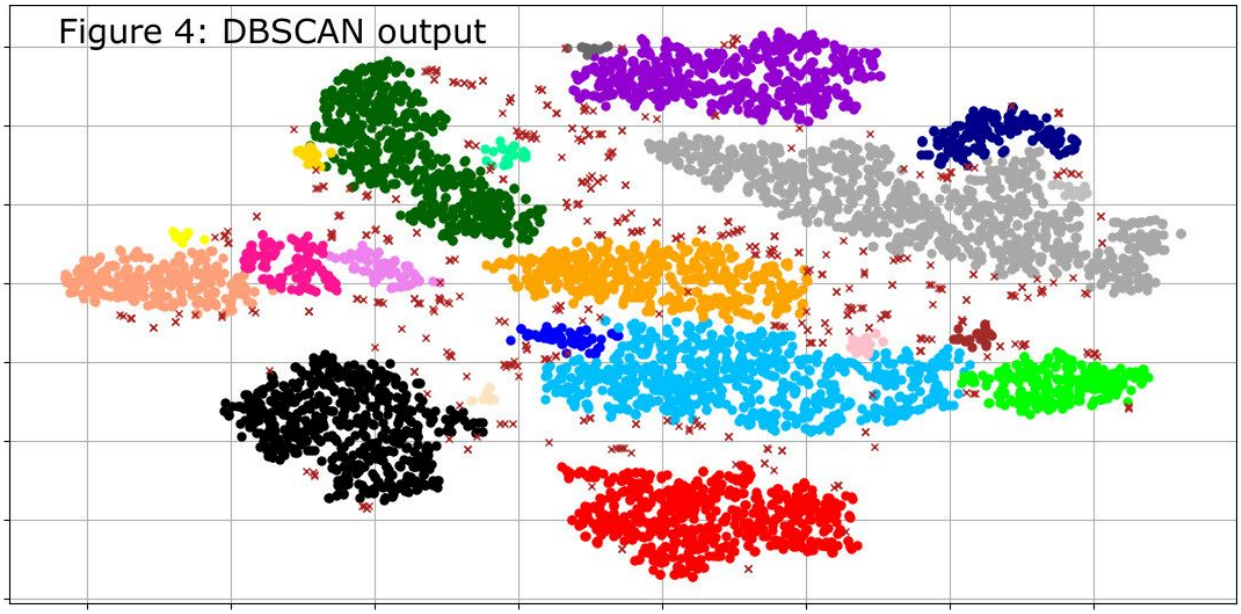
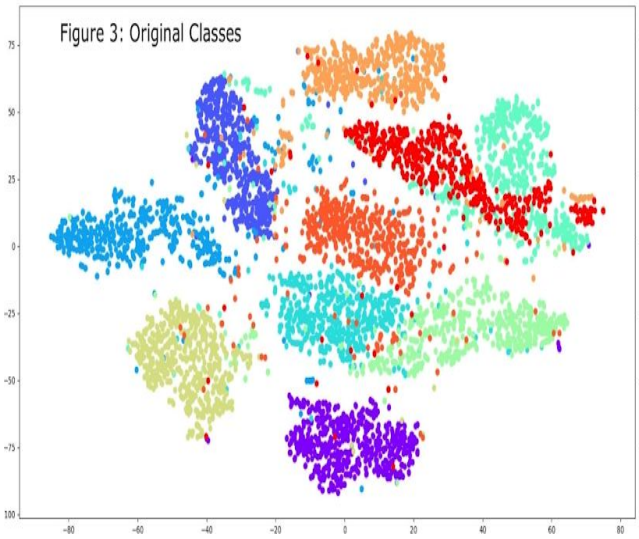
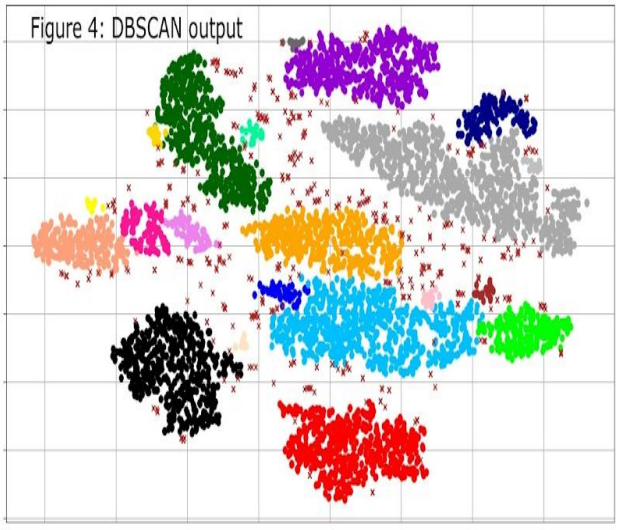


Figure 4: DBSCAN output

Part 3) Why are the clusters meaningful?

- Each colored cluster represents points with certain similarities.
- Original dataset had data points corresponding to 28x28 size images of the all the digits from 0-9.
- When we try to cluster the data points, we should be getting something like 10 clusters - where each cluster will represent a cluster of images of one digit.
- Since in our case we had the labels corresponding to each data point, hence we could just straightaway compare the results of clustering with the original class distribution. See following - original classes(Figure 3) vs final generated db scan output(Figure 4):





- But if we did not have those labels, then to analyze the clusters, what we can do is that we could calculate mean and variance of each cluster.

- When we calculated the mean and variance of each cluster and compared it with the original clusters, we found mean error to be very less implying that most the points clustered together belonged to the same cluster indeed.
- This suggested that the clusters obtained are meaningful indeed.

Part 4) Why did the approach worked?

- First of all we tried to use k-means algorithm to divide the dataset into clusters, but as the dimensionality of the data was quite high, k-means doesn't work well in these conditions as it took a lot of time to generate clusters.
- Also any distance-based similarity measure converges to a constant value between any given examples at a very large number of dimensions.
- Hence we tried to use PCA, but it wasn't helpful either due to a different reason - all the clusters got mixed into each other because of dimensionality reduction along an axis which maximised variance.
- So instead of PCA we used TSNE algorithm for dimensionality reduction. TSNE algorithm works in a very different way - by minimizing the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data, it focuses to preserve the local distances of the high-dimensional data in some mapping to low-dimensional data.
- K-means has a disadvantage that it will always try to output a spherical shaped cluster. So unless the clusters are spherical, this algorithm won't give good results.
- Since in our case clusters were not spherical using k-means and k-medoids gave large mismatches.
- Hence dimension reduction by PCA and clustering by k-means didn't yield great results.

- DBSCAN is a density based clustering algorithm, it does a great job of seeking areas in the data that have a high density of observations, versus areas of the data that are not very dense with observations, taking advantage of the output produced by the TSNE algorithm. It is also great in handling the outliers and detecting arbitrary shaped clusters very well.
- Hence we used dbscan algorithm to cluster our dataset.
- Hence the TSNE dimensionality reduction approach along with DBSCAN algorithm gave us the best results among the algorithms that were discussed.