HW3: Iris and Digits clustering using K-Means

Details:

Name: Aniket Pandey
Miner2 Username: 414
GMU Id: apandey7
K-Means - Iris

- Rank: 12
- Public Score: 0.95

K-Means - Digits

- Rank: 01
- Public Score: 0.84

Steps to run the code:

1. Download KMeans_Iris.ipynb and KMeans_Digits.ipynb file
2. Open JupyterLab on Anaconda Navigator or Google Colab Notebook
3. On the menu bar click File -> Open -> Filename
4. Keep the KMeans.py and test data file in the same directory
5. One by one run each cell in the notebook

Introduction:

To correctly label the data, we tried K-Means clustering algorithm. The algorithm identifies '*k'* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The *'means'* in the K-Means refers to averaging of the data; that is, finding the centroid.

Solution:

We implemented K-Means++ variation of K-Means to label our data in the correct cluster. Here are the steps to follow for K-Means++ algorithm.

Step 1: Pick the first centroid point (C_1) randomly.

Step 2: Compute distance of all points in the dataset from the selected centroid. The distance of x_i point from the farthest centroid can be computed by

$$d_i = max_{(j:1 \mapsto m)} ||x_i - C_j||^2$$

d_i: Distance of x_i point from the farthest centroid

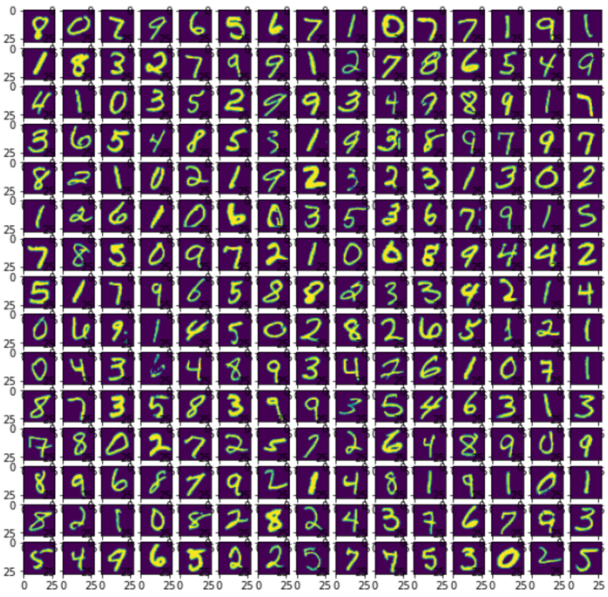
m: Number of centroids already picked

Step 3: Make the point x_i as the new centroid that is having maximum probability proportional to d_i.

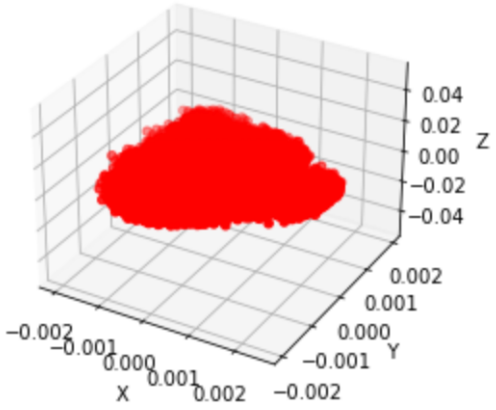
Step 4: Repeat the above two steps till you find k-centroids.

Process/Approach:

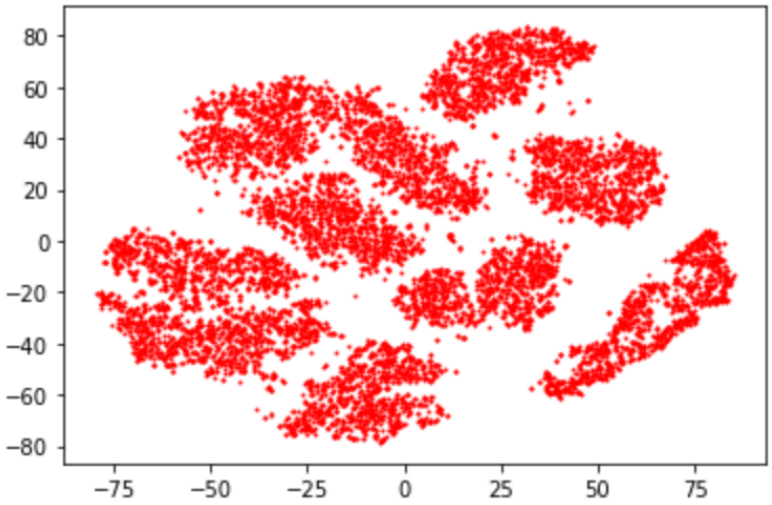
1. Import necessary packages and libraries. Following libraries are used in the classifier code:
   - Numpy for working mathematical operations on arrays
   - Pandas for Dataframe
   - sklearn.decomposition for PCA
   - sklearn.preprocessing for normalizing the data
   - sklearn.manifold for T-SNE
   - Matplotlib for plotting graphs
2. Read the test data using Pandas.
3. Visualize the current raw data to see the data we are about to cluster.



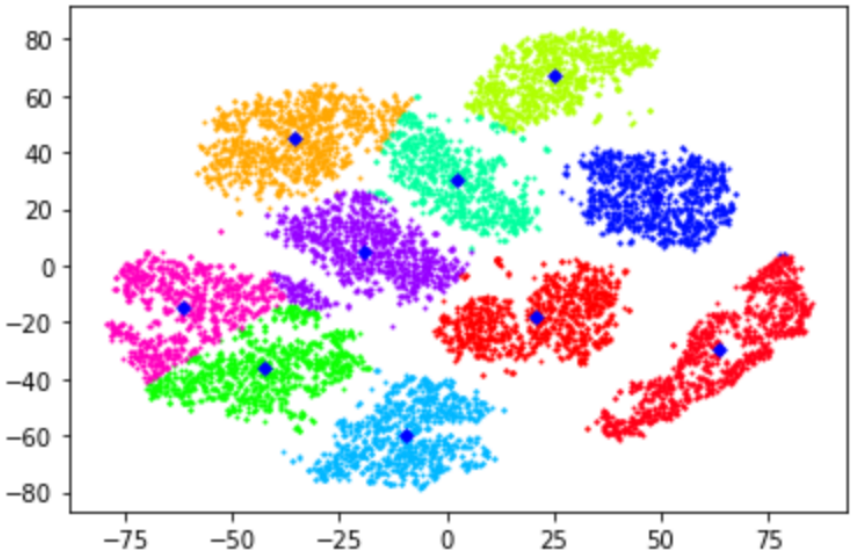
4. Normalize the data to bring all the data to a common scale.
5. By using Principal Component Analysis, transformed the data and performed dimensionality reduction.



6. Again, performed dimensionality reduction using t-SNE method. t-distributed Stochastic Neighbor Embedding also called nonlinear dimensionality reduction. What this means is that algorithm allows us to separate data that cannot be separated by any straight line. It is mostly used to understand high-dimensional data and project it into low-dimensional space like 2D or 3D.



7. Apply K-Means algorithm with 100 iterations on the prepared data.
8. The data gets categorized into their respective cluster.



9. Write the predicted output in a file using Numpy savetxt().

Conclusion:

Successfully implemented K-Means clustering algorithm using Python and labelled the digits image data to their correct cluster. The model can be used to correctly label the data.