CS6510: Applied Machine Learning

Assignment 3

Saksham Mittal

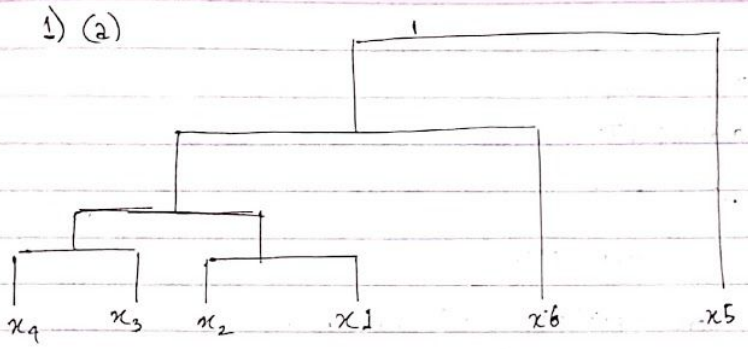
18.04.2019

CS16BTECH11032

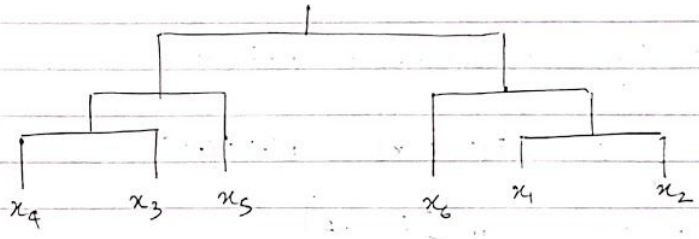
Theory Questions

Hierarchical Clustering

1) (a)



(b)



complete

(c) The initial step for link clustering is different from single link clustering as where AB and F are grouped together by distance (AB, F) = distance (B, F) = 0.61
So, we would like distance (AB, CD) = distance (A, D) to be less than 0.61,
So it can be 0.59.

Also, we want distance (ABCD, F) = distance (C, F) = 0.93 to be minimum so that ABCD, F are grouped together.
so it can be 0.65.
Dendogram will remain same after these changes are performed.

(2) Principal Component Analysis:

(a) $E[X] = \frac{E[a]}{M} * \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \frac{\mu}{M} * \frac{1}{p}$

$V = E[(X - E[X])(X - E[X])^T]$

We know,

$$E[XX^T] = E[a\delta_i \; a'\delta_j]$$

$$= E[aa'\delta_{i,j}]$$

$$= \begin{cases} \mu^2 & \text{if } i = j \\ 0 & \text{other cases} \end{cases}$$

over all K, for expectation divide by M.

$$E[XX^T] = \frac{\mu^2}{M} \times (I)$$

on substituting, the covariance matrix →

$$V = \frac{\mu^2}{M} I - 2\,E[X]\,E[X]^T + E[X]\,E[X]^T$$

$$= \frac{\mu^2}{M} I - \frac{\mu^2}{M^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & & & 1 \\ \vdots & & & \vdots \\ 1 & & & 1 \end{bmatrix}$$

(b) The covariance matrix →

$$\left(\left(\frac{\mu^2}{M}\right) I - \left(\frac{\mu^2}{M^2}\right) \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \right) \underline{1}$$

where $\underline{1}$ is a vector of ones.

For $\Rightarrow \quad \left(\frac{\mu^2}{M}\right)\underline{1} - \left(\frac{\mu^2}{M^2}\right) \underline{1} \times M \times 0 = 0$

Hence $\underline{1}$ is an eigenvector with value 0.

$(1, 1, \ldots 1)$

we can also represent the covariance matrix as:

$$\left(\frac{\mu^2}{M^2}\right)\left(MI - \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \right) = C \begin{bmatrix} M-1 & -1 & -1 & \cdots & -1 \\ -1 & M-1 & & & \vdots \\ \vdots & & M-1 & & \vdots \\ & & & \ddots & \vdots \\ -1 & -1 & \cdots & & M-1 \end{bmatrix}$$

where $C = \frac{\mu^2}{M^2}$

$$\therefore \quad |V - \lambda I| = \begin{bmatrix} M-1-\lambda & -1 & -1 & \cdots & -1 \\ -1 & M-1-\lambda & \cdots & & -1 \\ \vdots & & & & \vdots \\ -1 & -1 & \cdots & & M-1-\lambda \end{bmatrix}$$

$$R_2 \to R_2 - R_1, \quad R_3 \to R_3 - R_1 \ldots$$

$$= \begin{bmatrix} M-\lambda-1 & -1 & \cdots & -1 \\ \lambda - M & M-\lambda & 0 \cdots & 0 \\ \lambda - M & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ \lambda - M & 0 & \cdots & M-\lambda \end{bmatrix}$$

$$= \left(1 + \frac{-1}{(M-\lambda)} \times M \right) \times (M-\lambda)^n$$

$$= -\lambda (M-\lambda)^{n-1}$$

So, eigenvalues are 0 (which was of $\frac{1}{}$) or the repeated value of M.
Hence proved!.

(c) As almost all the eigenvalues are same, PCA <u>will not</u> perform well. So, removing dimensions will result in a huge loss in variance.

Programming Questions

[Python version used in assignment = 2.7.15]

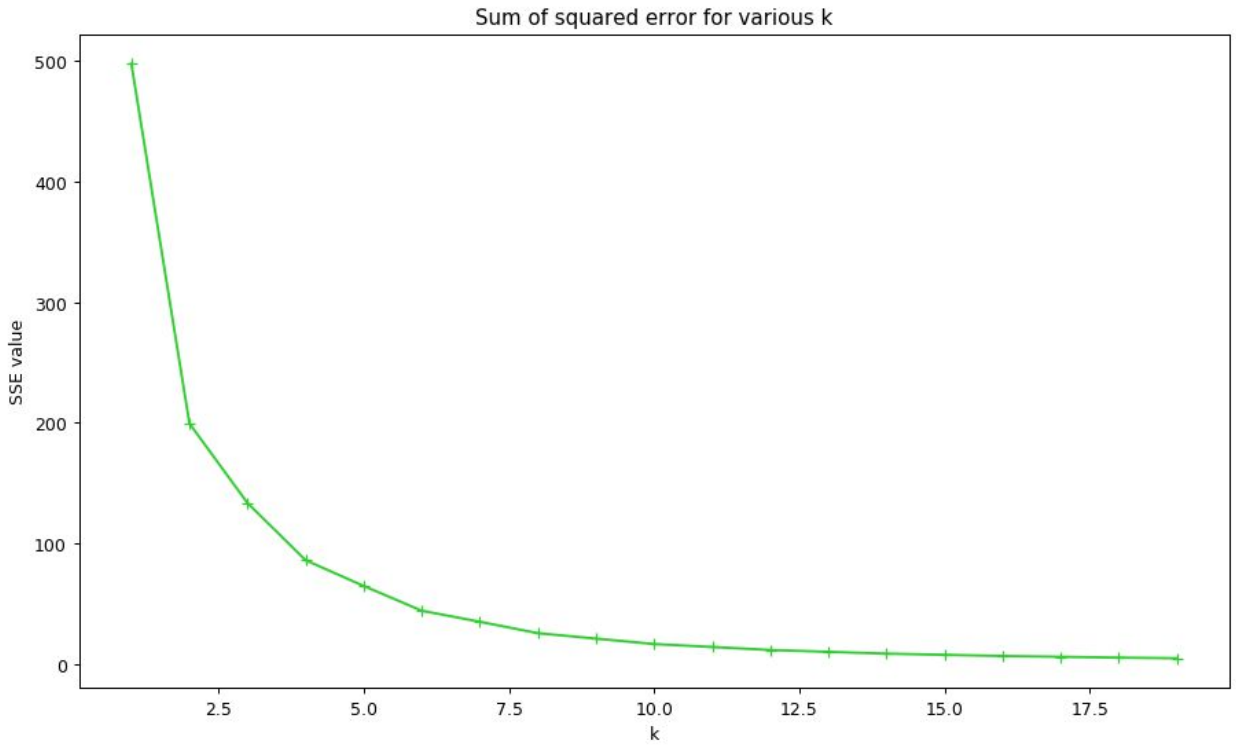
3. Clustering:

The code is provided in **3-clustering.ipynb**

a) The code uses numpy for mathematical calculations and loading of data and matplotlib for graph plotting. I used kMeans algorithm from the sklearn library for finding the number of clusters in dataset1. I used the **elbow method** to determine the optimal number of clusters for k-means clustering.

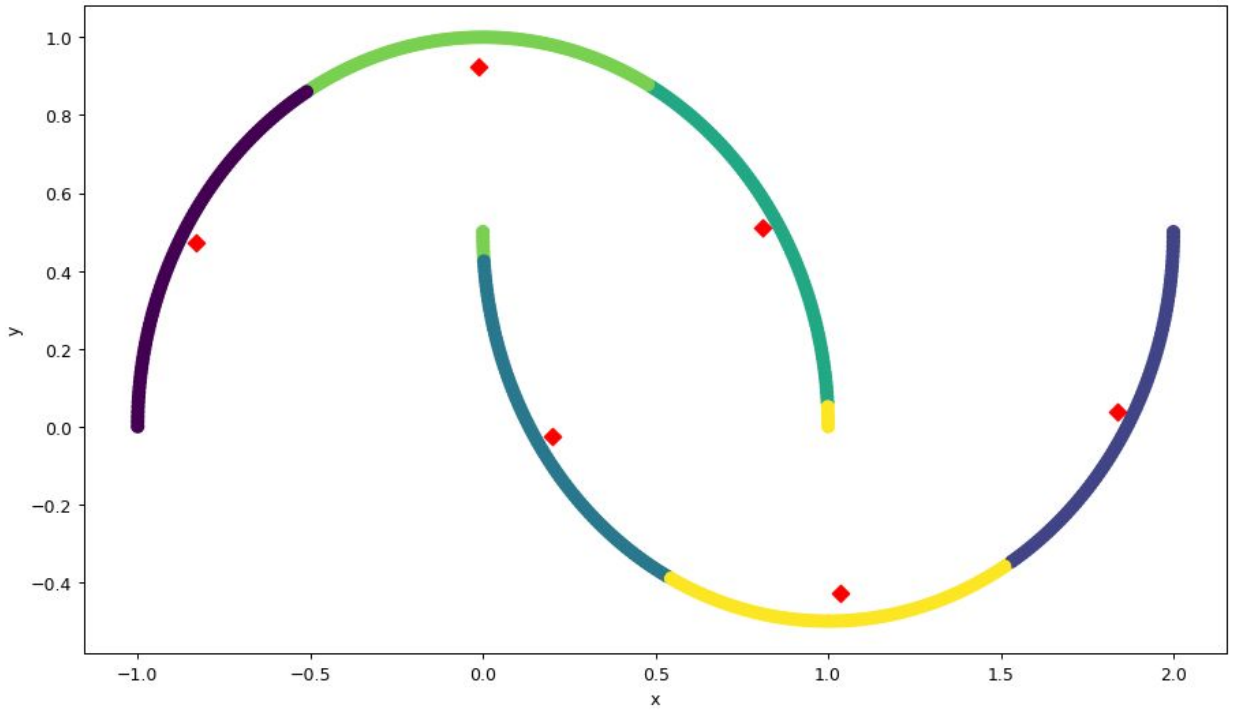
The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k (say, k from 1 to 20 is used for dataset1), and for each value of k calculate the sum of squared errors (SSE). We choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k.

The elbow curve I got for the dataset1 is:



As we can see, the elbow is formed as k = 6.

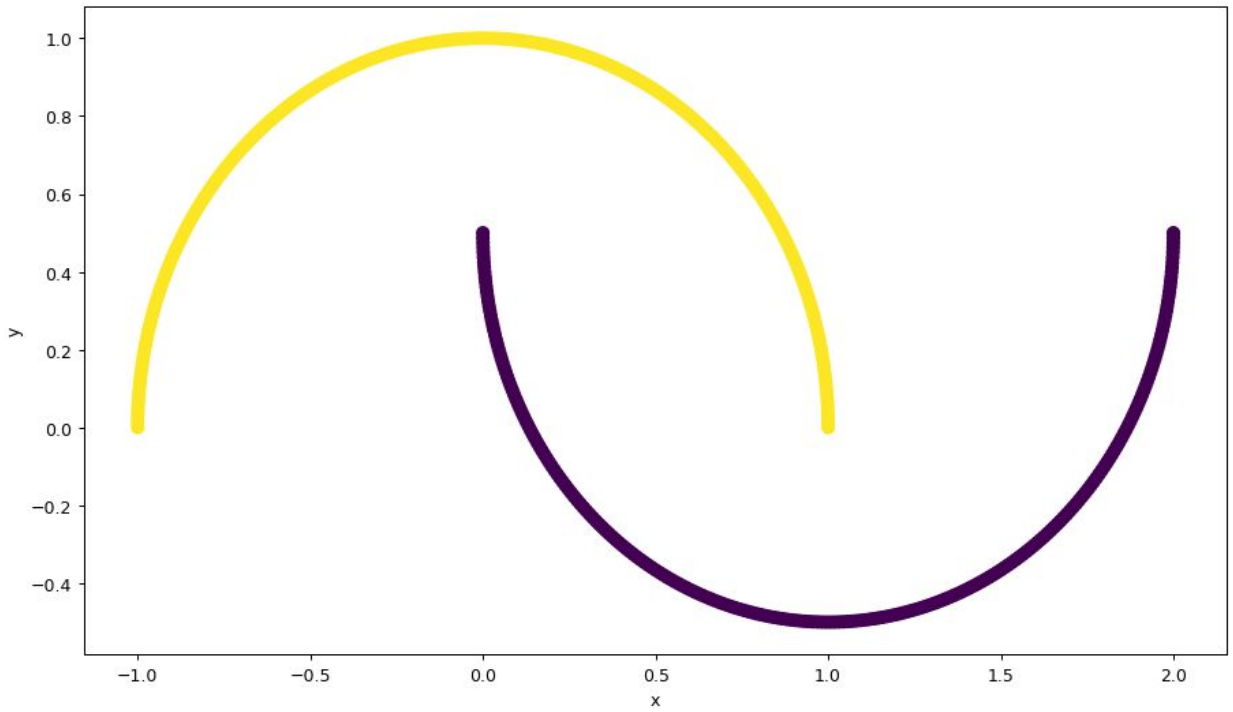
The plot for kMeans for clusters = 6 is:



b) I have implemented my own method my_dbscan() which runs for all points and assigns them to clusters.

The labels are set to -1 for outliers.

The plot on using Dbscan on dataset1:

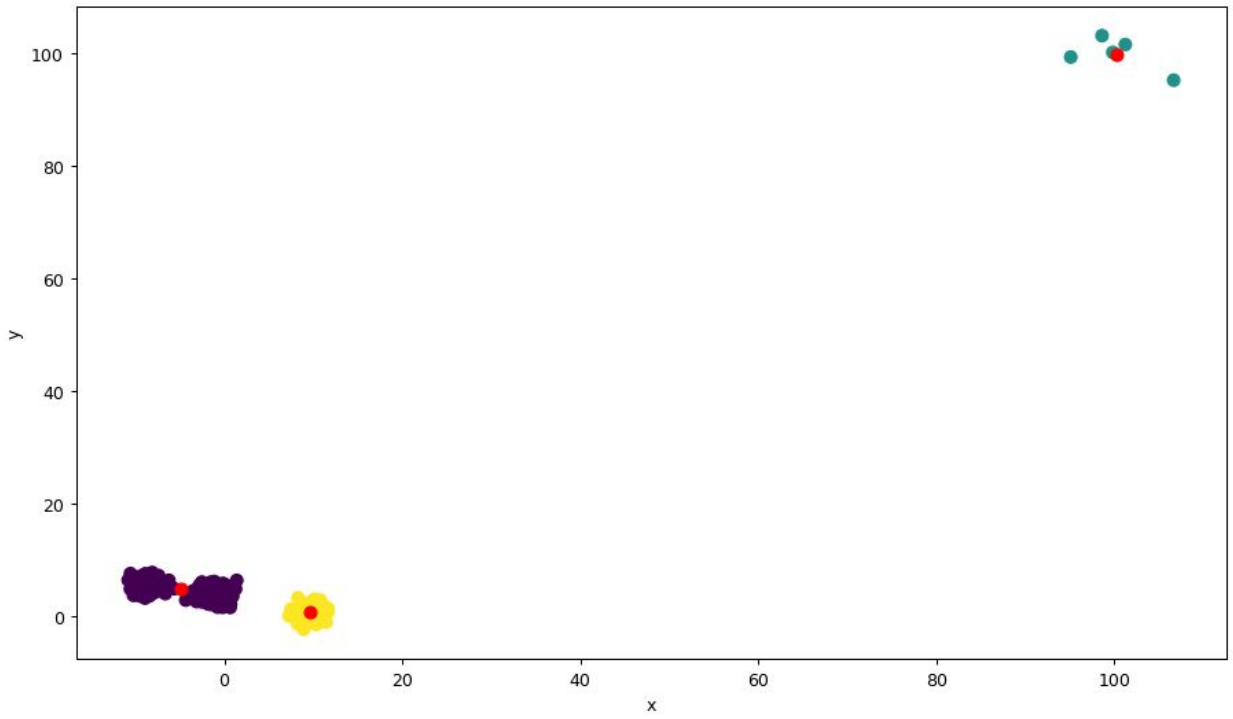


c) As we can see from the plots, the dbscan forms 2 clusters of dataset1 while kMeans algorithm forms 6 clusters. The reason can be attributed because of the difference in how they are implemented under the hood.

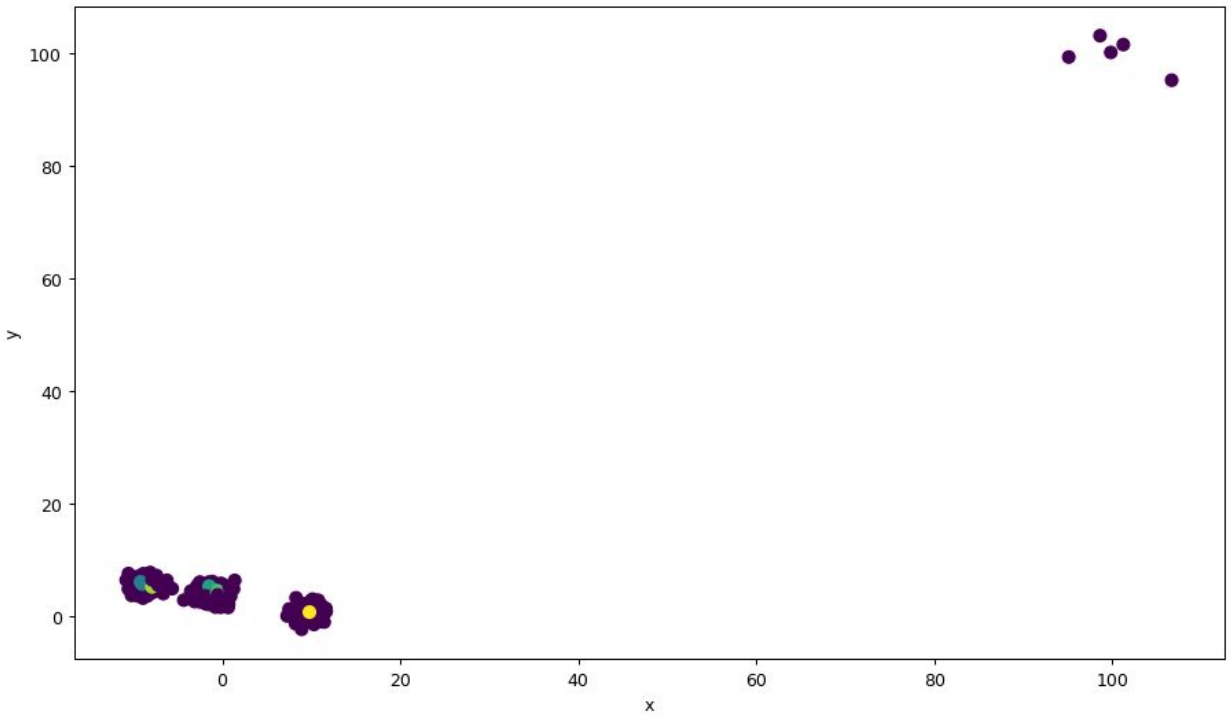
   Dbscan is based on concept of reachability, which tries to find the neighbours within a given radius of a point. While, kMeans uses distance concept, finding the clusters from a central-clustering point. It also requires us to mention the number of clusters beforehand, which we may not know. But, if the clusters are present in different densities in the data, the Dbscan may not perform good.

d) For dataset2, the following graphs are obtained:

   Using kMeans(number of clusters = 3 as given in question):



   Using Dbscan:



In the first graph, as we know kMeans forms clusters from a central-clustering points, we can see the red dots as the centers of the clusters.

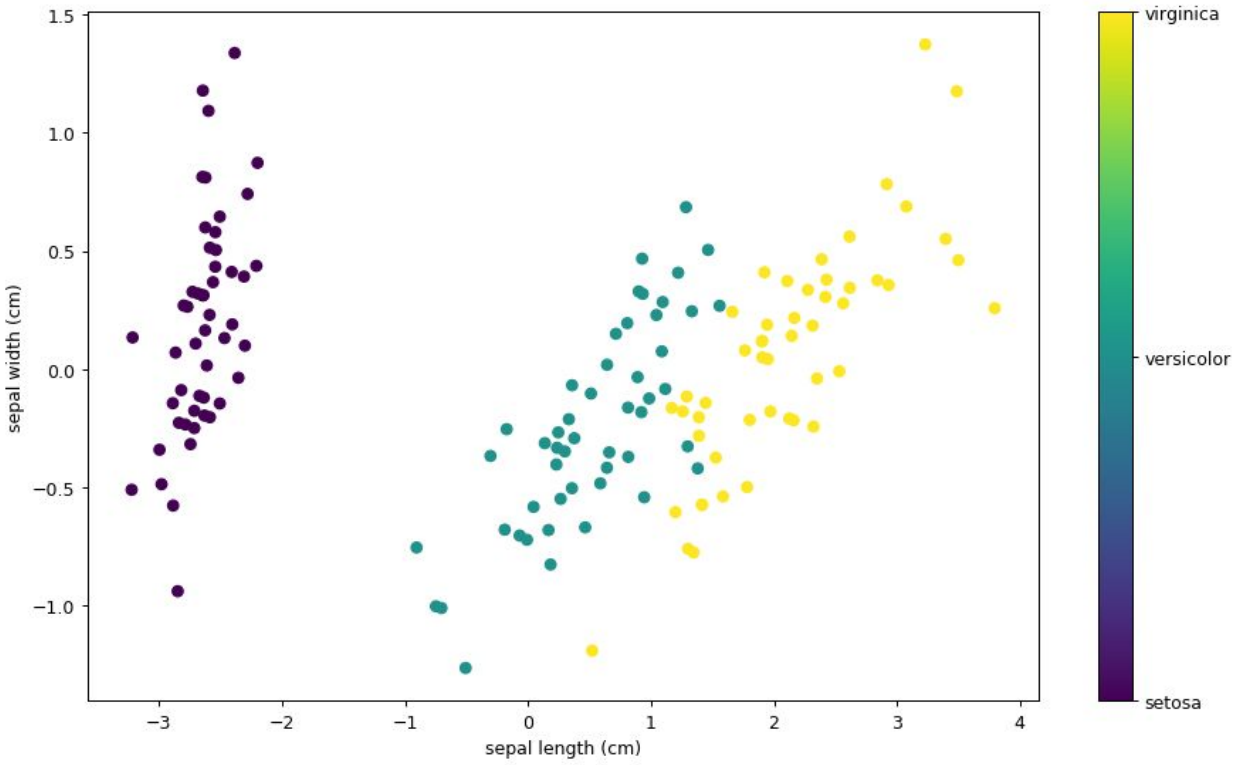
Pros and Cons of Dbscan and kMeans:

- Kmeans is a faster algorithm than Dbscan
- We don't need to explicitly provide the number of clusters in the Dbscan algorithm. This is particularly useful for data in which visualization is not easy.
- Dbscan doesn't work well with data in which clusters are of different densities.
- As the distance calculation in kMeans is done commonly using Euclidean distance, calculating it for higher dimension data may be not easy.
- Dbscan helps to find even clusters of arbitrary shapes.

4. Dimensionality Reduction:

The code is provided in 4-a.py, 4-b.py, and 4-c-tsne.py and 4-c-pca.py respectively for parts a), b), and c). Also, the plots are provided in the same directory.

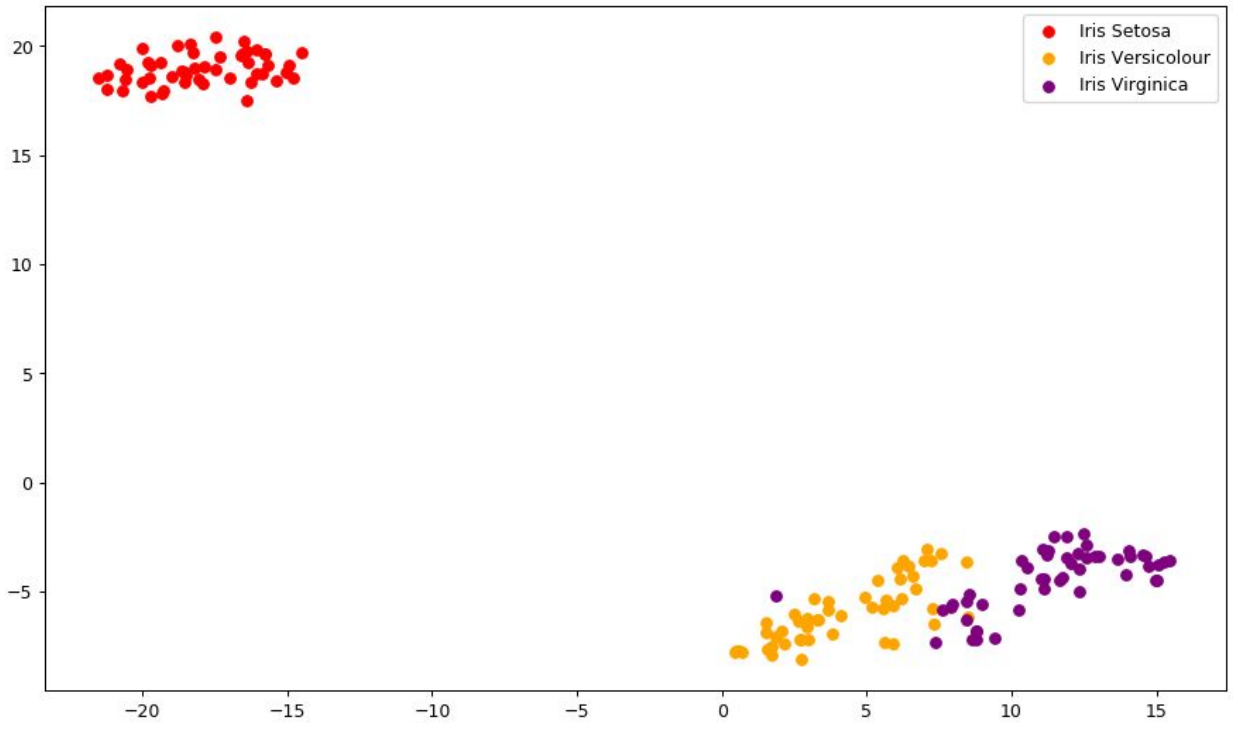
a) The iris dataset is loaded using the sklearn library. The plot obtained for this dataset is:



PCA performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized.

The clusters formed are loosely bounded.

b) The plot obtained on using t-SNE is:



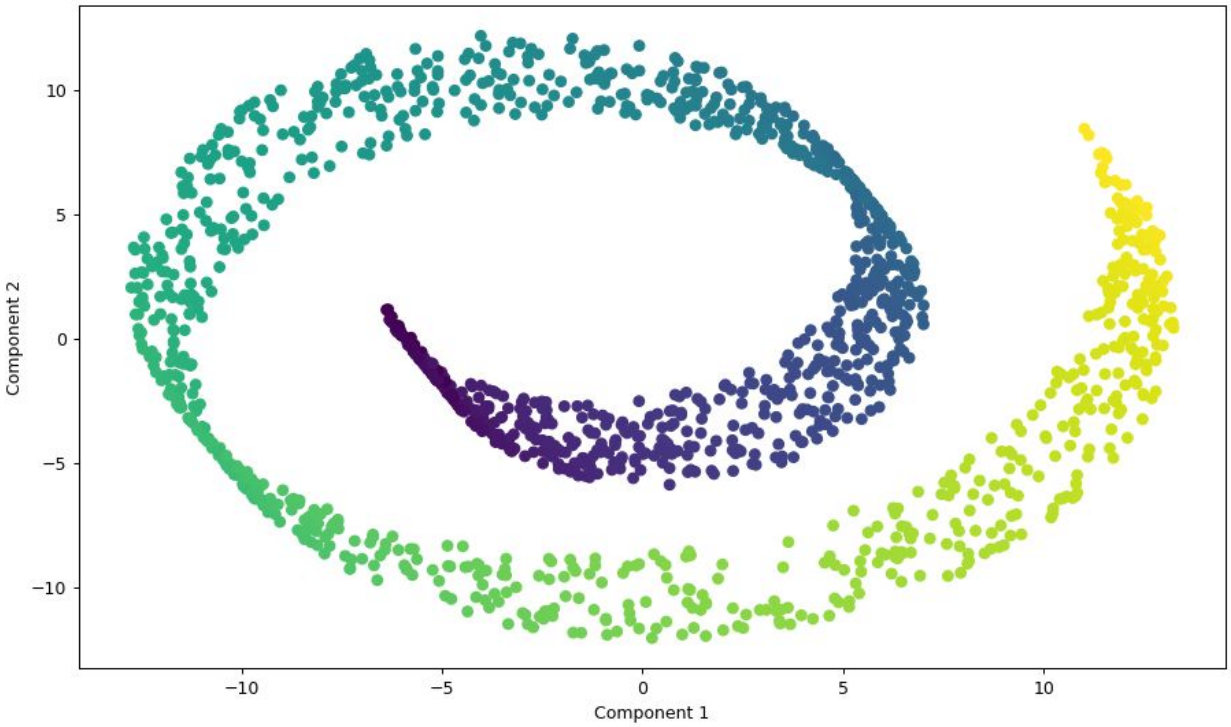
As we know, t-SNE is a probabilistic technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

Whereas PCA is a linear feature extraction technique.

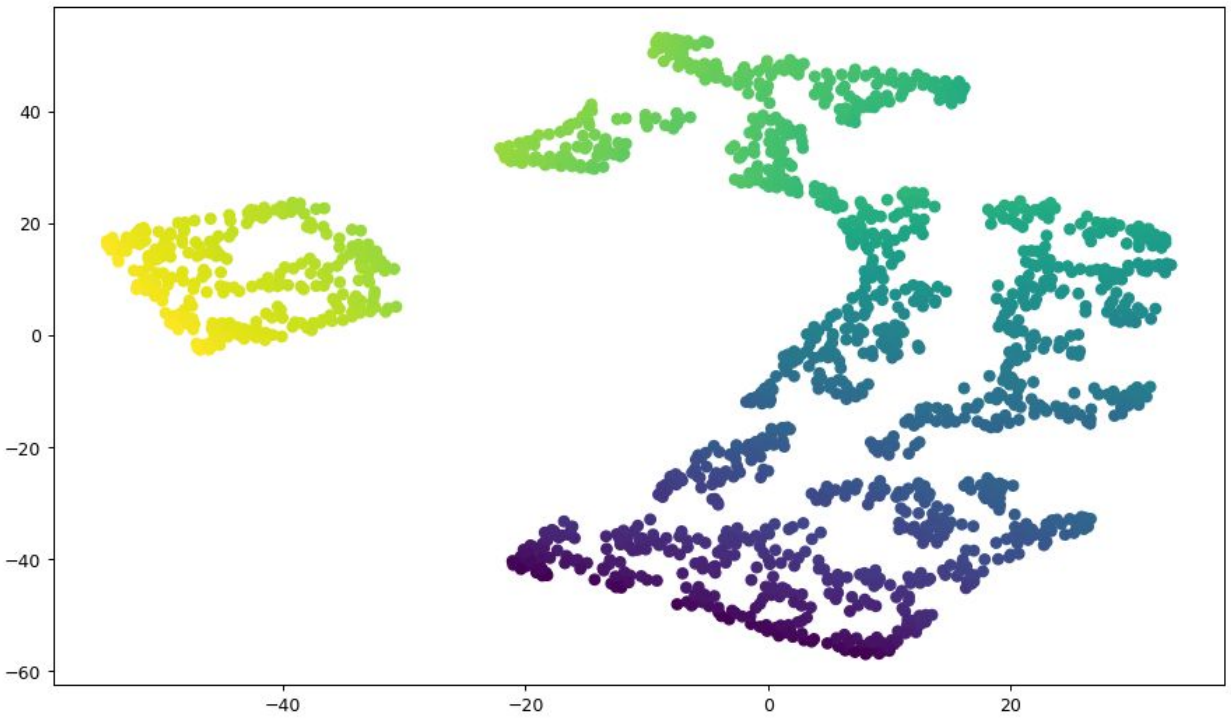
The differences and similarities can be summed as:

- Both of the algorithms helps to reduce the features while preserving the relationship between the features.
- t-SNE is more computationally expensive than PCA, and the difference is more evident when the data is of the order of millions.
- PCA is a mathematical technique, while t-SNE is a probabilistic one.
- Sometimes in t-SNE even with the same hyperparameters may produce different results hence multiple plots must be observed before making any assessment with t-SNE, while this is not the case with PCA. (This can be attributed to the probabilistic nature of the t-SNE algorithm)
- PCA is a linear algorithm, it will not be able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that.

c) The PCA plot for SwissRoll data is:



The t-SNE plot for the SwissRoll dataset:



The SwissRoll dataset is non-linear. Hence, the PCA doesn't give proper clustering, while t-SNE is well suited for the visualization of high-dimensional datasets which have non-linear relationships between features.