CS5691: Pattern Recognition and Machine Learning

Assignment 1:

Unsupervised Learning: PCA and K-means Clustering

Shrivarshan K, MM20B058

1. Principal Component Analysis (PCA)

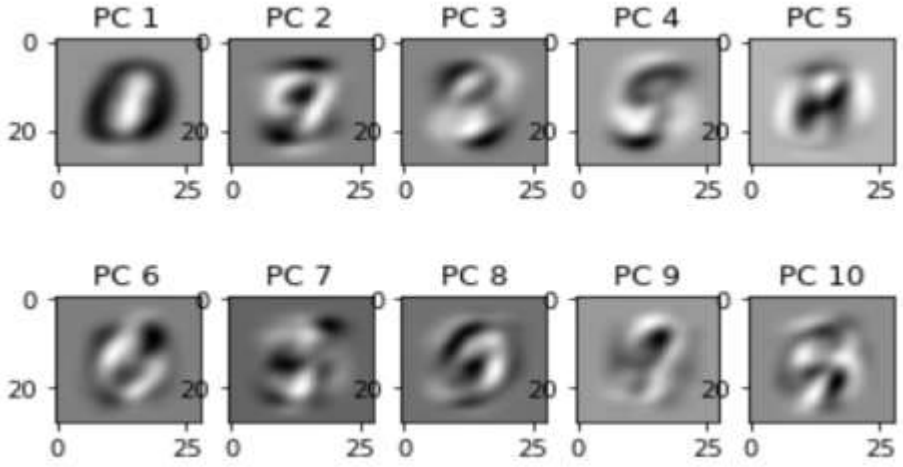
Question – (i):

The given dataset is MNIST dataset which consists of handwritten digits. Each data point is a grayscale image of size 28*28. The dataset is downloaded using **mnist.load_data** function and the training data which contains 60000 datapoint is considered for the analysis. Each data point is converted into a vector of dimension 784 = 28*28 and with these vectors as columns, X matrix of dimension 784*60000 is formed and centered.

After centering, the Covariance matrix C = X^TX/n of dimension 784*784 is formed. Then Eigen value decomposition of the matrix C is done to find the principal components. For this **eigh** function from **Numpy** library is used which gives eigenvalues and eigenvectors of the matrix in a sorted manner.

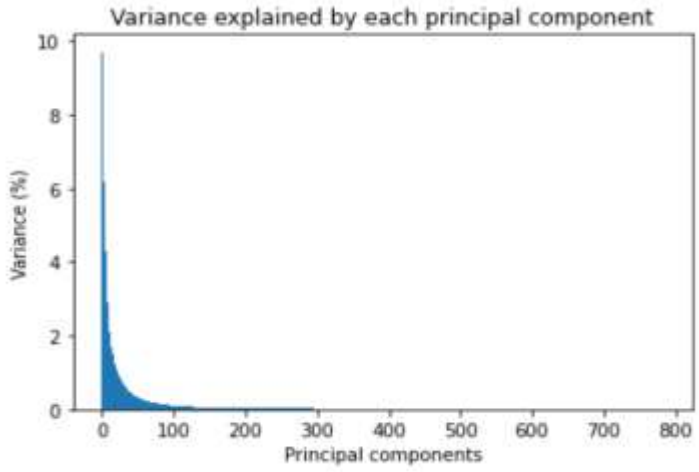
Images of principal components:

The co-variance matrix is of very high dimension, so the number of eigenvectors is in the order of several hundreds. The images of top 10 eigenvectors are generated and plotted using **plt.imshow** function from **Matplotlib** library.



Variance explained by each of the principal components:

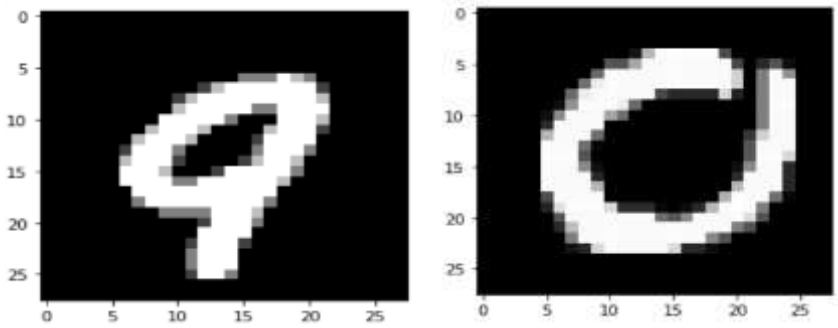
The variance explained by an eigenvector or principal component is nothing but the eigenvalue associated with the eigenvector. So the percentage variance explained by each vector is essentially the eigenvalue associated with the eigenvector over sum of all eigenvalues. This math is done for all the 784 eigenvectors and a bar plot of variance vs principal components is plotted using **plt.bar** function from **Matplotlib** library.



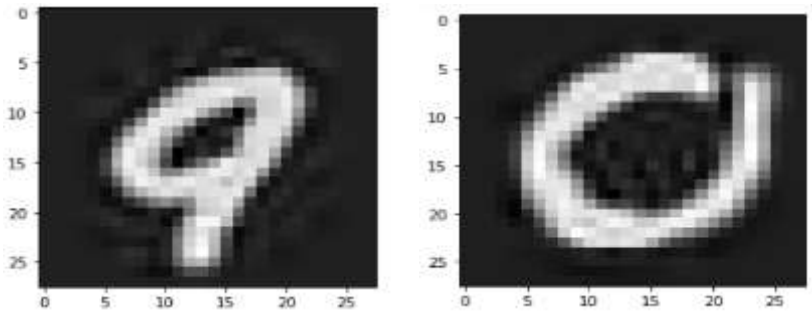
The maximum percentage variance explained is around **9.7%** which is by the first principal component.

Question – (ii):

In order to reconstruct the dataset we need to choose the number of principal components to consider. We need to find a d, such that, the linear combination of top d eigenvectors explains 95% of the variance of the data. We found out that **d = 154.** So the top 154 principal components are considered to regenerate the dataset. As the number of datapoints is 60000 and it is difficult to regenerate the while dataset, a parameter j is introduced. It is essentially the index of the datapoint which we are interested to regenerate. A few samples are shown below.



Random samples from MNIST dataset

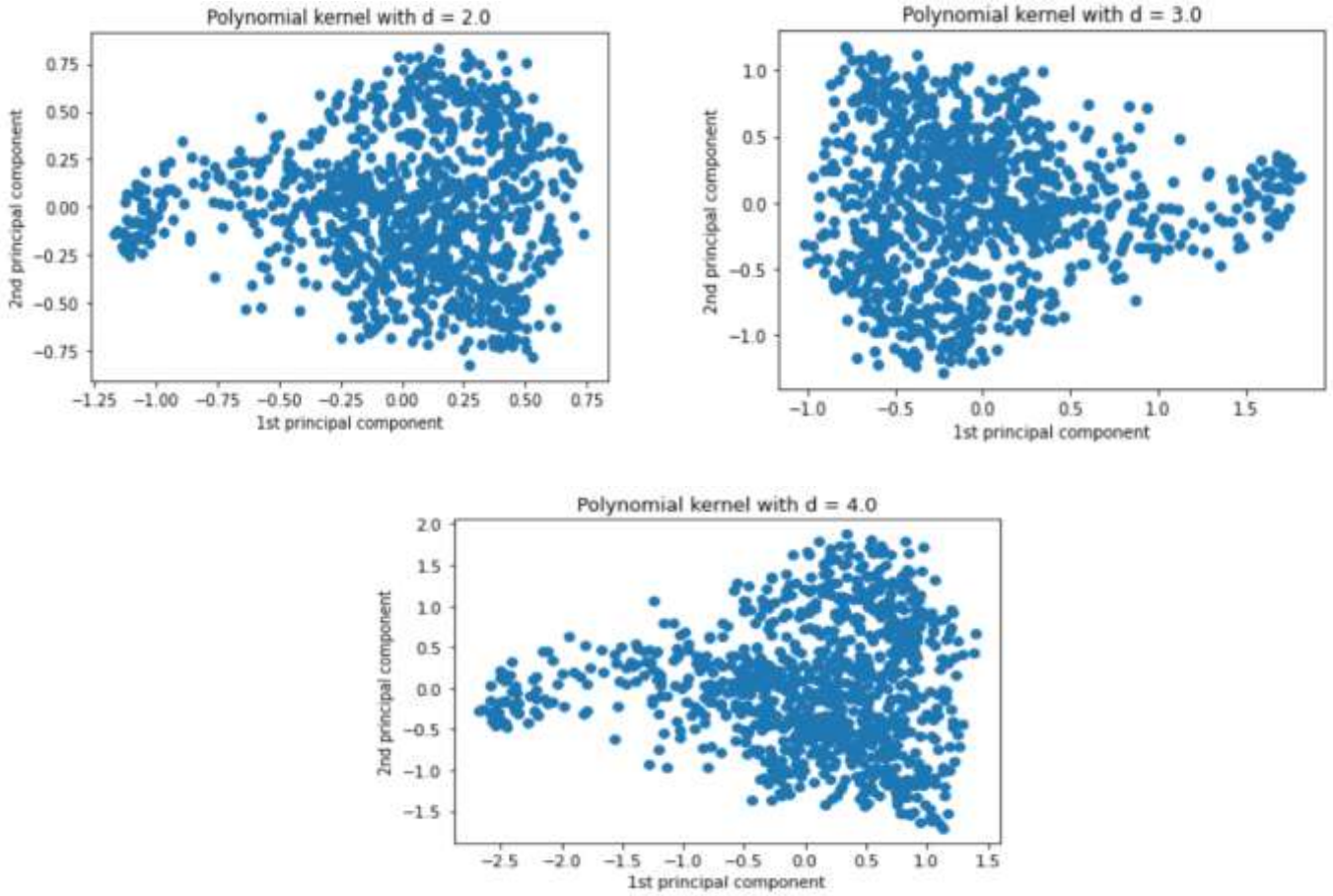


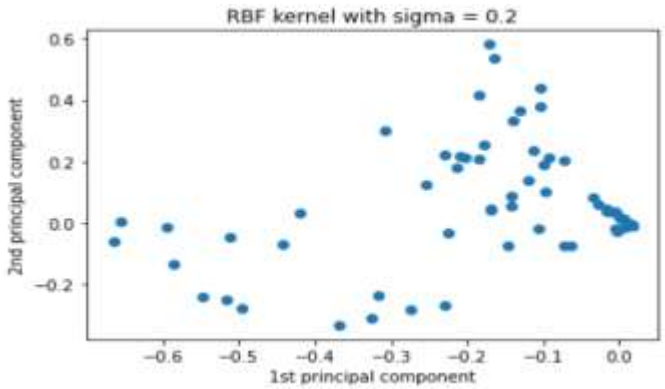
Regenerated images using top 154 principal components

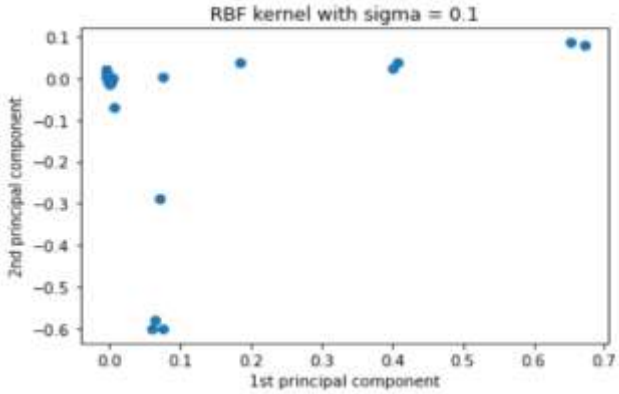
Question – (iii):

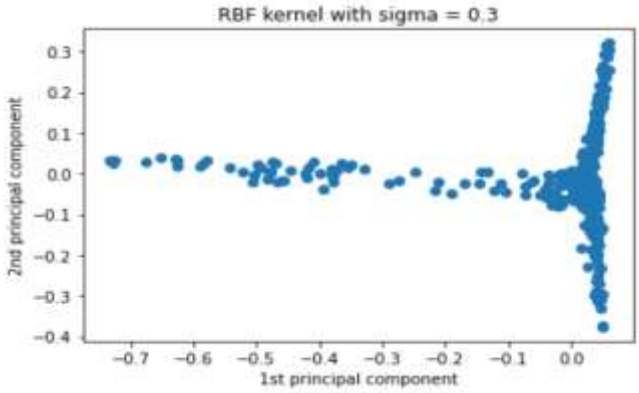
For this question two functions, **get_kernel_comps** and **show_kernel_pca** are written. Since it was very time consuming to find the two dimension representation of 60000 datapoints, a subset of 1000 datapoints is considered. The get_kernel_comps function computes the kernel matrix based on the choice of kernel which can be given as input by the user. Then the kernel centering step is done. Then eigenvalues and eigenvectors of the centered kernel matrix is calculated and using which the alpha-ks are calculated. This function return the proxy of each datapoint in two dimensional space as we are considering only two principal components.

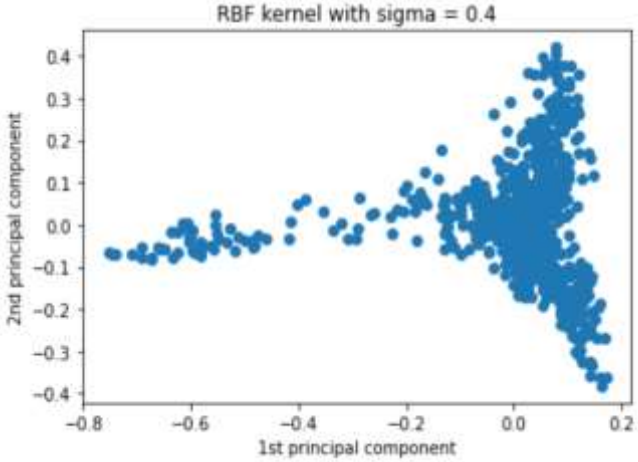
The two dimensional representation of each datapoint using Polynomial and RBF kernel are listed below:

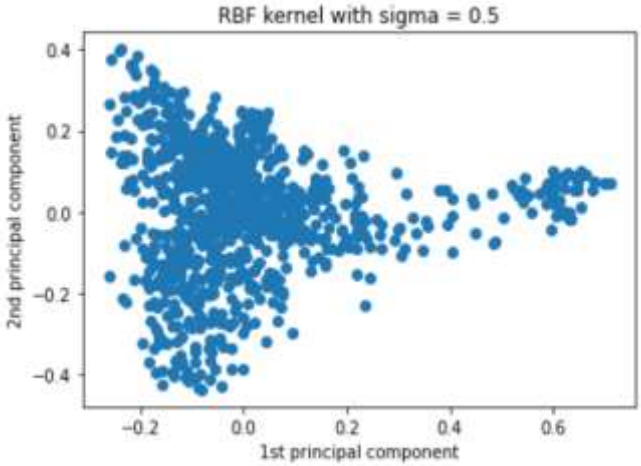


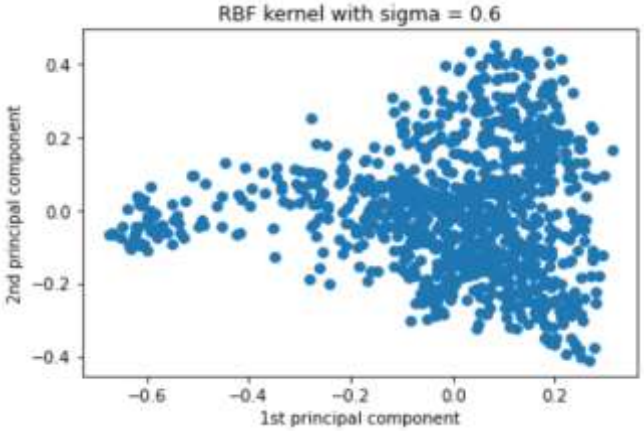

RBF kernel with sigma = 0.2
2nd principal component
1st principal component

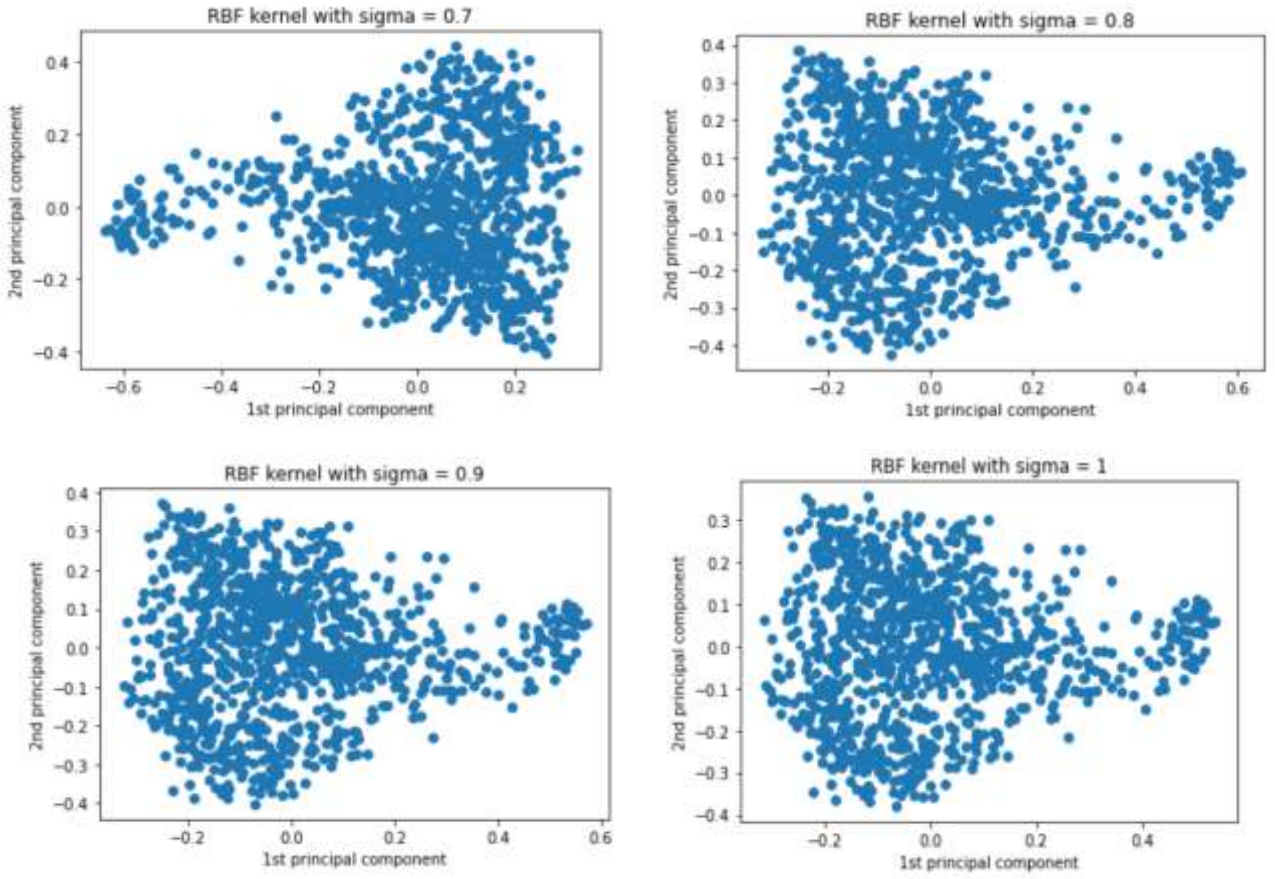

RBF kernel with sigma = 0.1
2nd principal component
1st principal component


RBF kernel with sigma = 0.3
2nd principal component
1st principal component


RBF kernel with sigma = 0.4
2nd principal component
1st principal component


RBF kernel with sigma = 0.5
2nd principal component
1st principal component


RBF kernel with sigma = 0.6
2nd principal component
1st principal component



Plots obtained by polynomial and RBF kernel

Question – (iv):

Variance explained by each of the kernel is given in the following table:

Kernel	Parameter	Variance
Polynomial	d = 2	0.14669
Polynomial	d = 3	0.12537
Polynomial	d = 4	0.10736
RBF	sigma = 0.1	0.00244
RBF	sigma = 0.2	0.00713
RBF	sigma = 0.3	0.01833
RBF	sigma = 0.4	0.03864
RBF	sigma = 0.5	0.06285
RBF	sigma = 0.6	0.08481
RBF	sigma = 0.7	0.10226
RBF	sigma = 0.8	0.11558

RBF	sigma = 0.9	0.12571
RBF	sigma = 1	0.13347

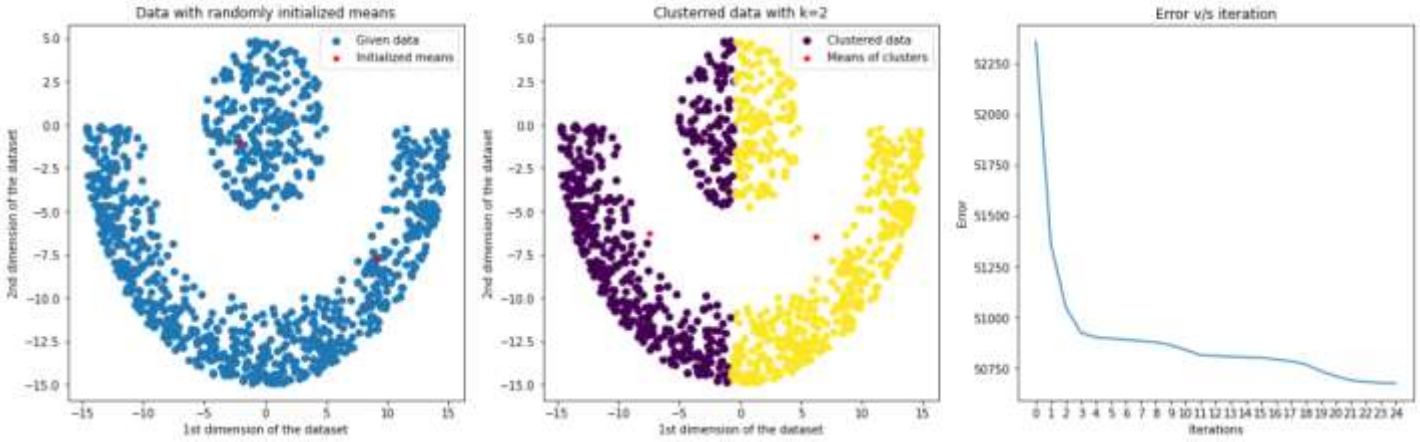
Based on the variance explained by each of the kernel the polynomial kernel with degree = 2 is the best one suited for this dataset.

2. K-means Clustering:

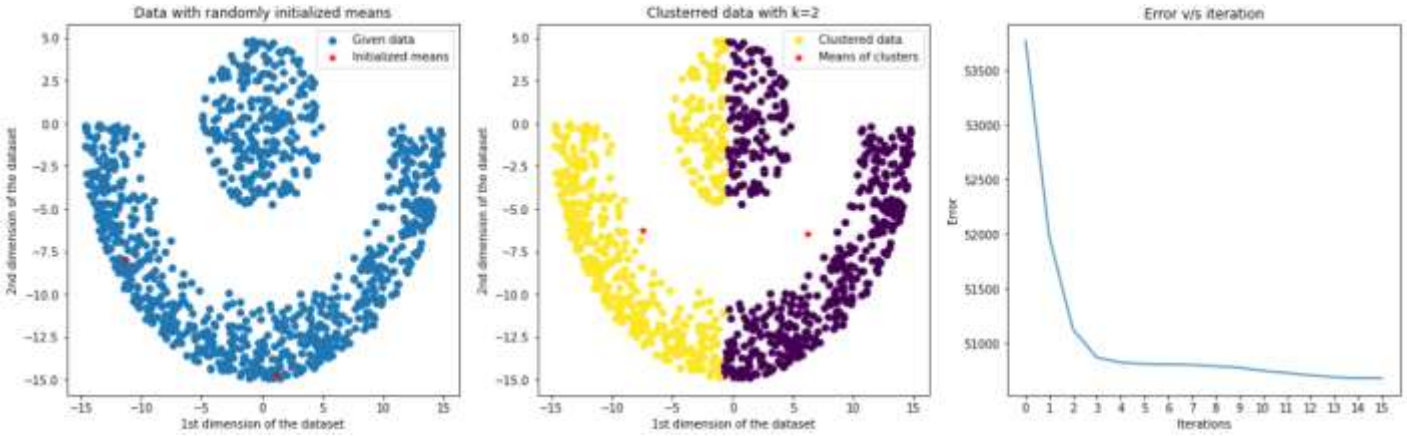
Question – (i):

The means are randomly chosen from the dataset using a function **sample** from **random** library. The following are the plots of clusters obtained and the respective error vs iteration for 5 random initialization of means.

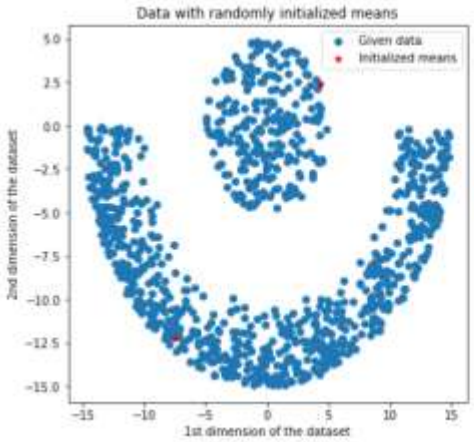
Random initialization 1:

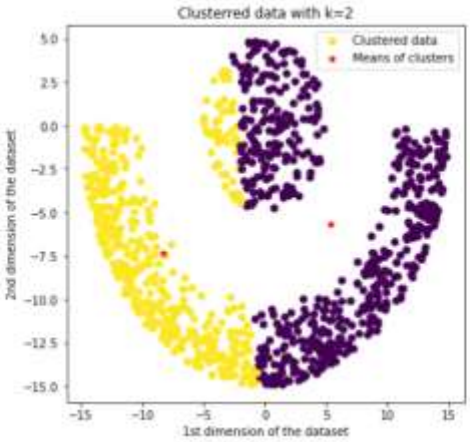


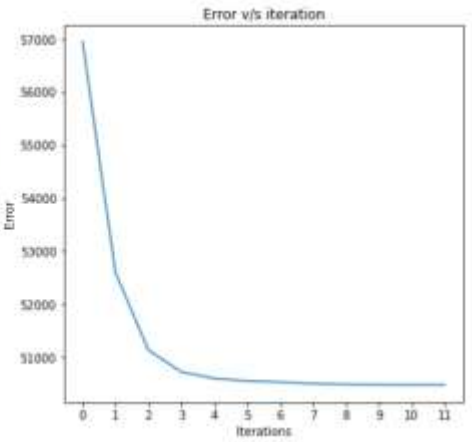
Random initialization 2:



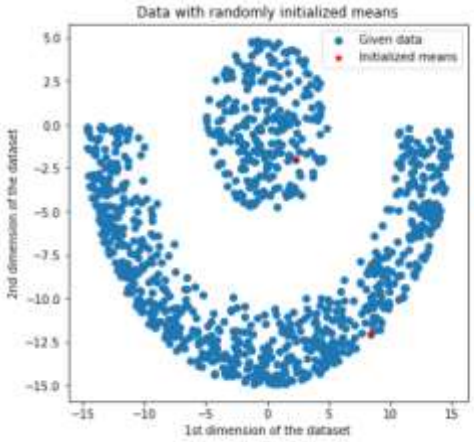
Random initialization 3:

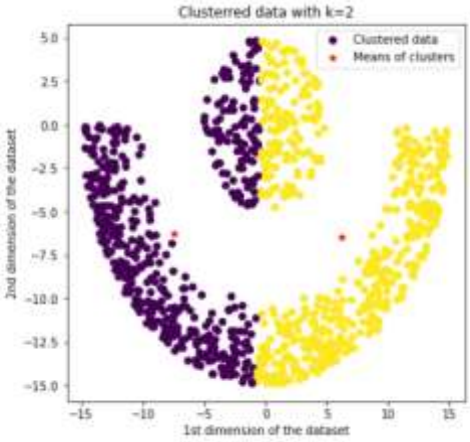


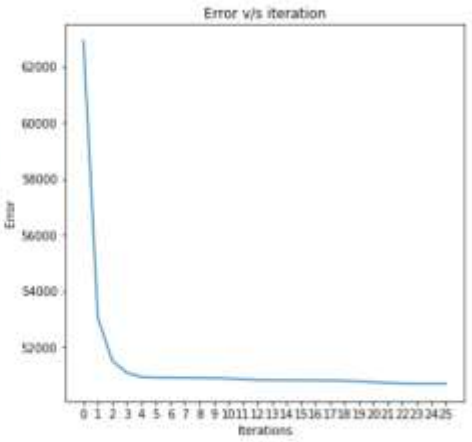




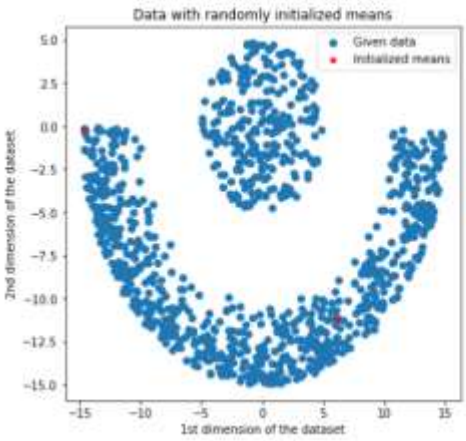
Random initialization 4:

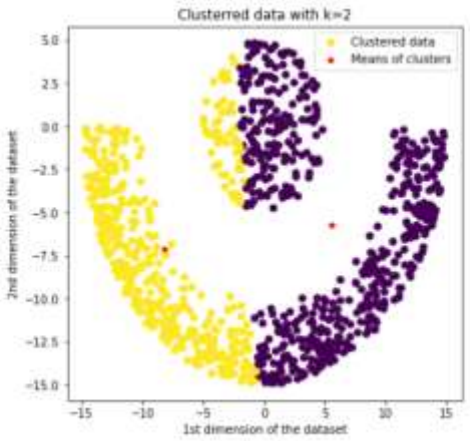


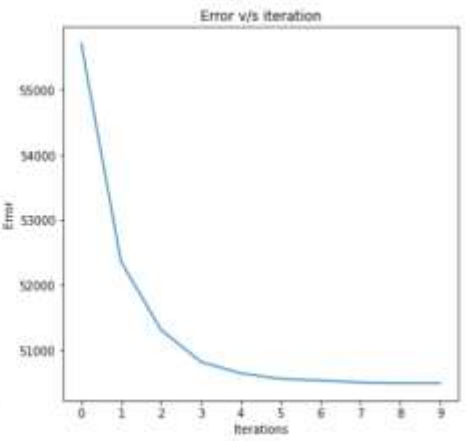




Random initialization 5:



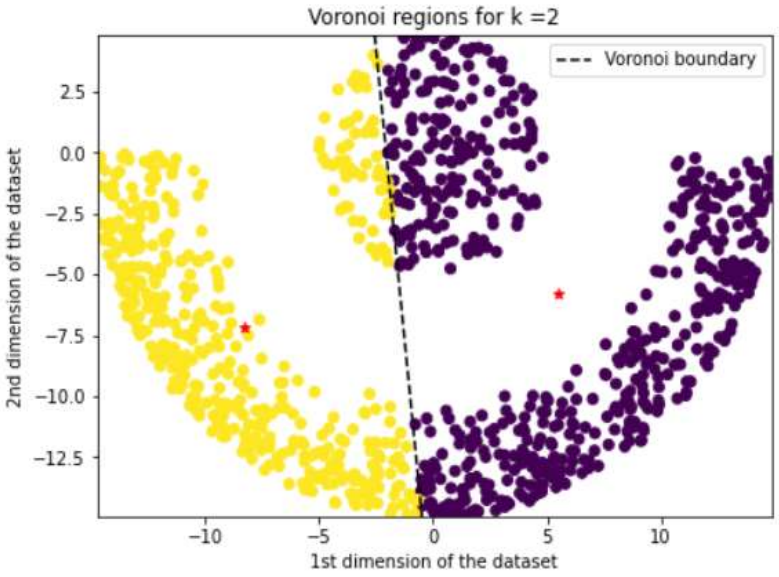




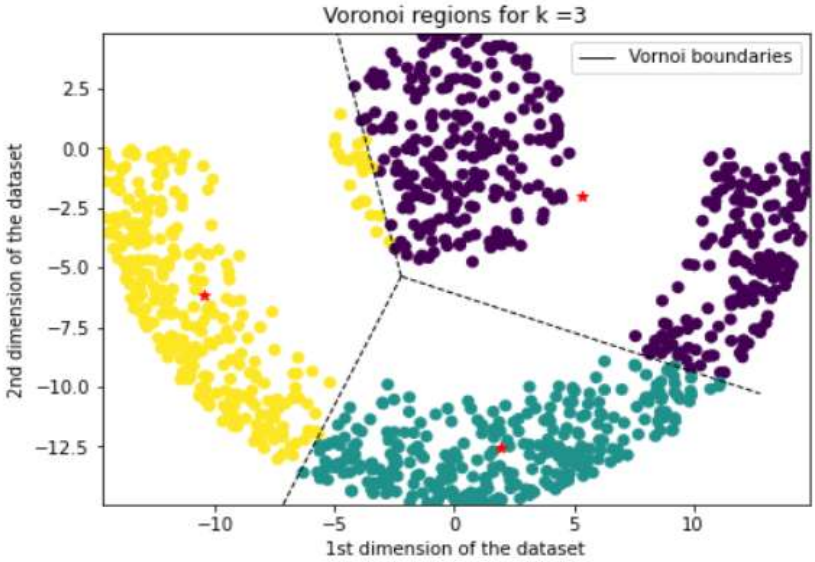
Question – (ii):

A random initialization is fixed and for each k, k number of means are taken from this set for initialization. Voronoi regions are plotted using the **voronoi_plot_2d** function from **scipy** library. The following are the Voronoi regions obtained for respective k.

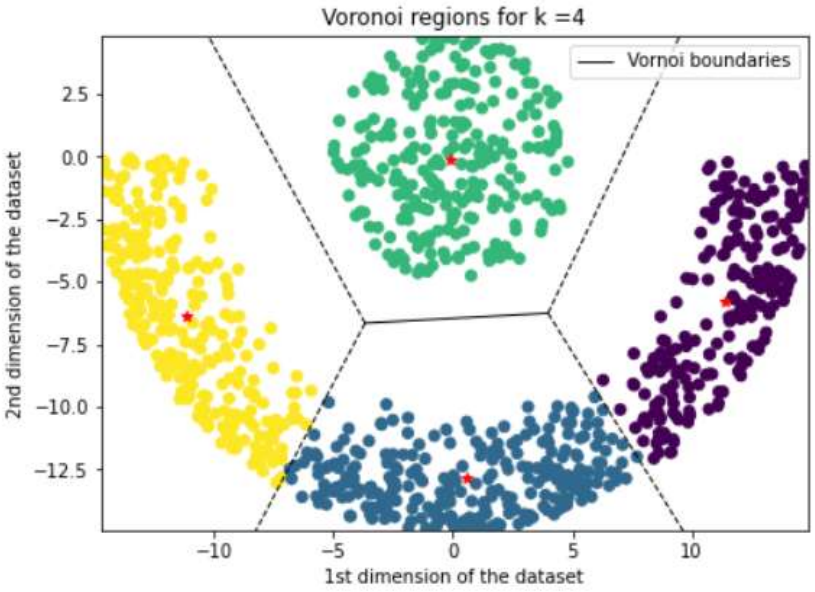
K = 2:



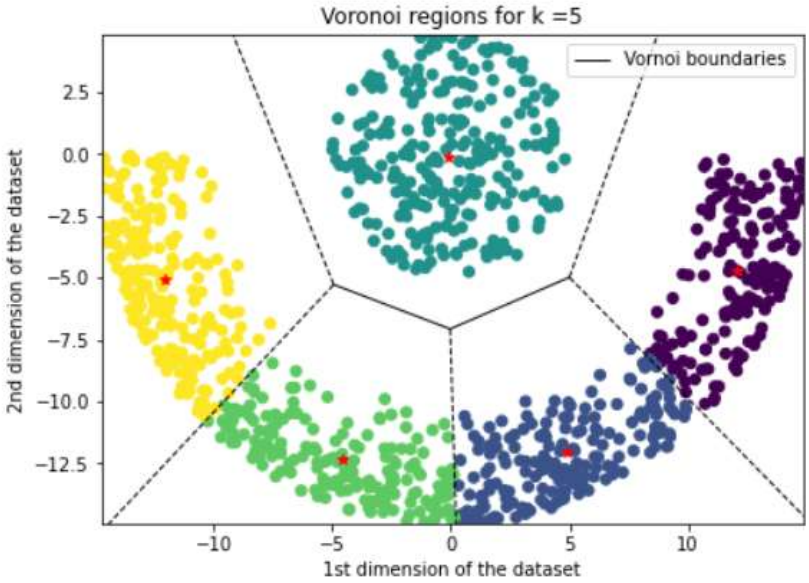
K = 3:



K = 4:

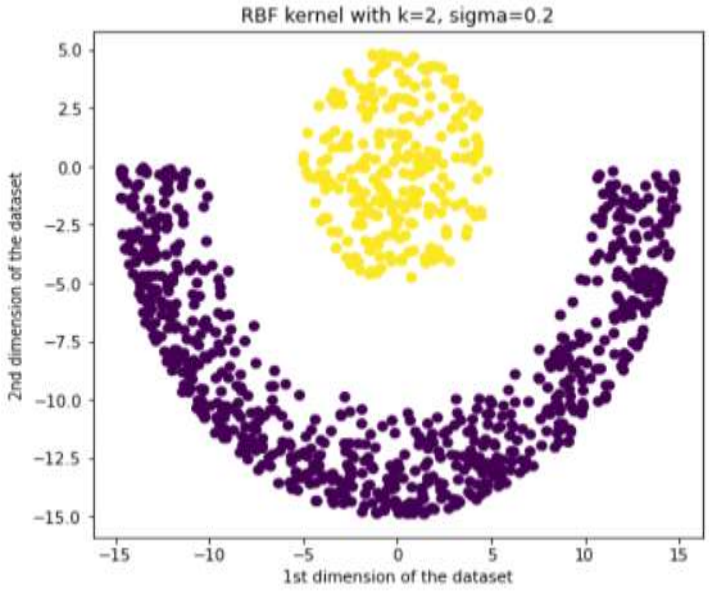


K = 5:



Question – (iii):

Polynomial kernel and RBF kernel with wide range of parameter values are tried. But the ideal clustering that we would like to get is not obtained. So a transformation called Laplacian transformation is used, and calculated L matrix. Then H-star matrix is created using the top k eigenvectors of L matrix and ran Lloyds. The result is the following.



Spectral Clustering using Laplacian transformation

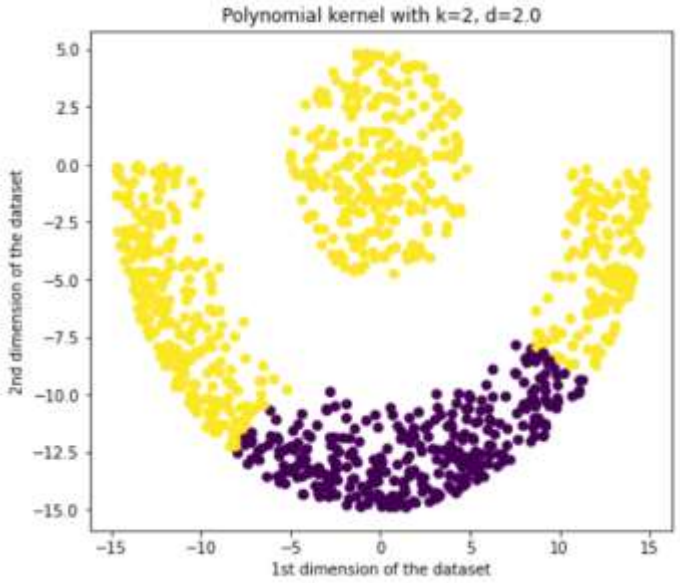
With reference to the above clustering, which is the ideal clustering, polynomial and RBF kernels are used without Laplacian transformation. The following are the clusters we got.

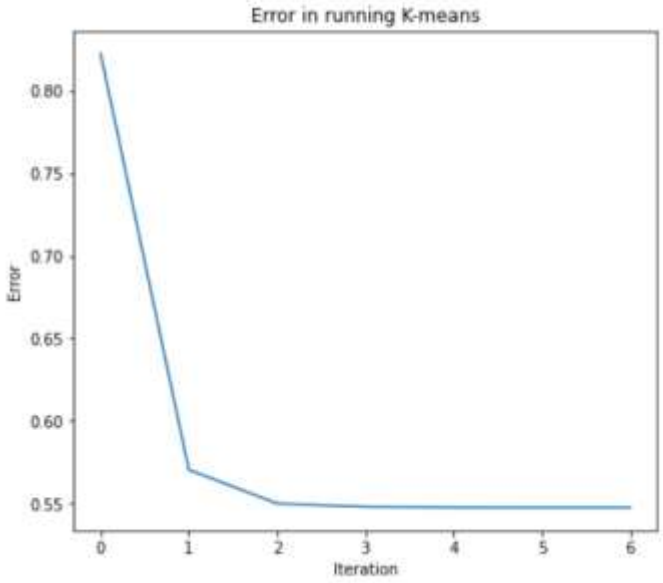
Polynomial Kernel:

The polynomial kernel with d = {2,3,4,5} were ran and got the following error values after convergence while running K-means with H-star matrix.

Degree	Saturation error
2	**0.54733**
3	0.77966
4	0.67039
5	0.88419

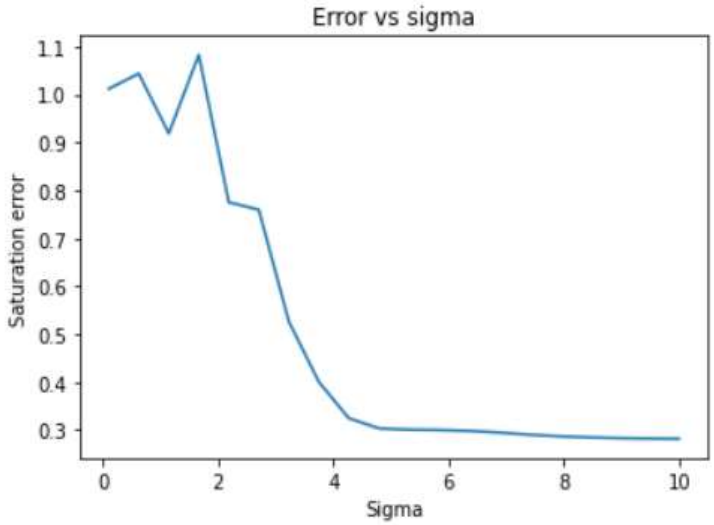
So the polynomial kernel with degree = 2 performs better among the polynomial kernels. The cluster obtained using the polynomial kernel with degree = 2 is as follows.



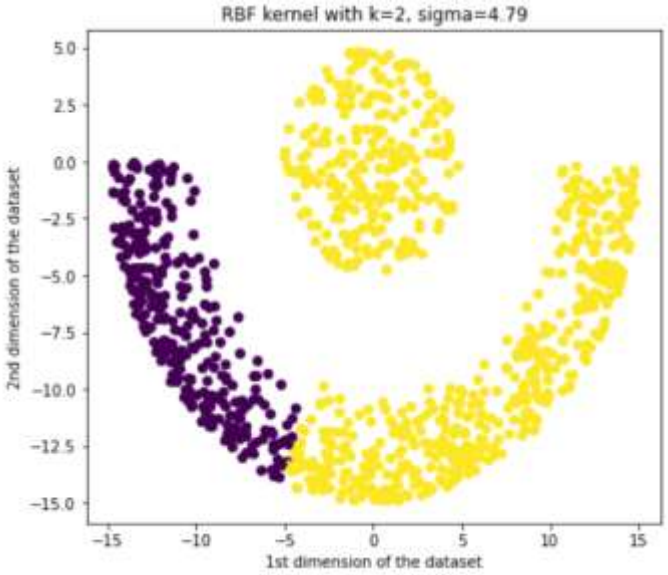


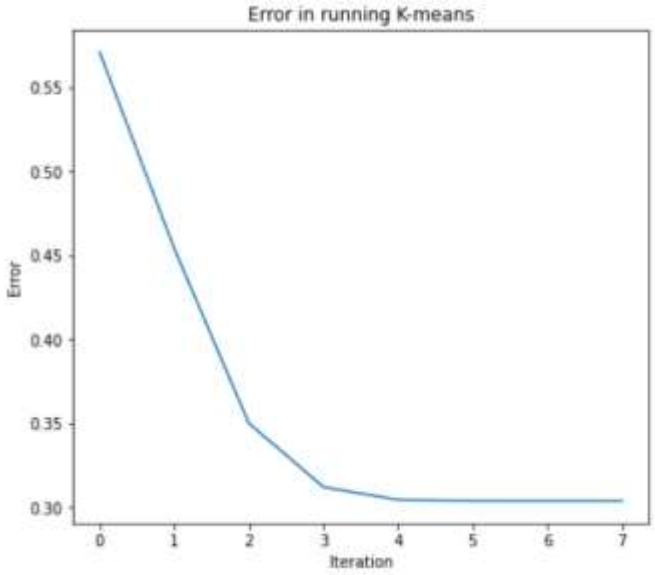
RBF kernel:

RBF kernel with sigma values ranging from 0.1 to 10 were ran and got the error values. But we know that as the sigma value increases, the error value decreases. So it would not be ideal to just compare using the error values. Thus the cluster regions obtained using each sigma value was observed and compared with the ideal clustering obtained using Laplacian transformation to get the better one.



From the above plot, we can say that the ideal sigma should be somewhere between 4 and 6 as until then it is the signal part of the graph and after that it is the noise part of the graph.

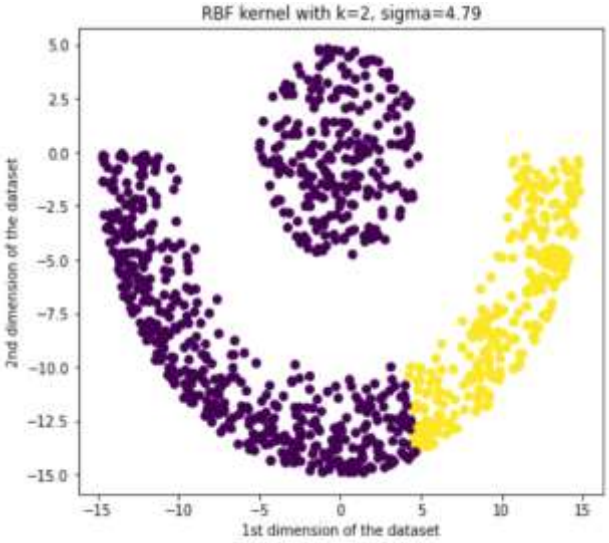




This RBF kernel with **sigma = 4.79** performs better with the saturation **error = 0.3.** On the whole, RBF kernel is suitable for the dataset than the polynomial kernel which is evident from the error values obtained.

Question – (iv):

The method of taking the maximum value of the eigen vector for each datapoint will essentially yield a clustering similar to the clustering yielded by RBF or polynomial kernel. This is because the datapoints having greater value of 1st eigenvector will eventually be in the same cluster and vice-versa. Lloyds algorithm on the H-star matrix also yields the same clustering. This is evident from the following plot.



Using RBF kernel with sigma = 4.79 also yielded a similar clustering. Thus this method is similar to performing kernel operation and running Lloyds on the H-star matrix.