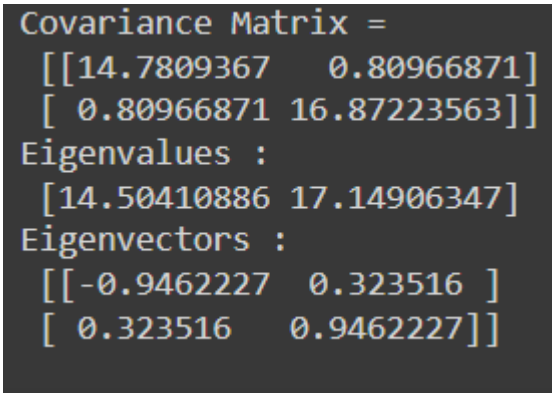
Question 1)

i) After running the PCA algorithm on the dataset I obtained two eigenvalues, lambda_one = 17.14906347 and lambda_two = 14.50410886. Please refer to the screenshot below -



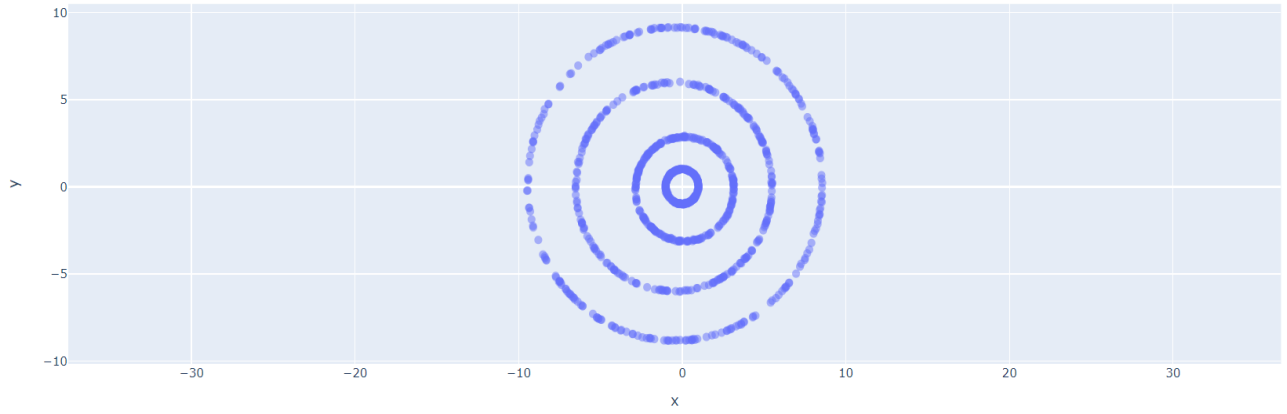
Here we have two principal components w_1 = [0.323516 0.9462227] and w_2 = [-0.9462227 0.323516]. We know that the eigenvectors of the covariance matrix give “good” directions and eigenvalues say how good the directions are. Also variance of the projected lengths on an eigenvector is equal to the corresponding eigenvalue. So the higher the eigenvalue the higher the variance along that direction. In this case lambda_one corresponds to the first principal component w_1 = [0.323516 0.9462227] and the variance of the projected lengths along w_1 is lambda_one = 17.14906347. Similarly, the variance of the projected lengths along w_2 = [-0.9462227 0.323516] is lambda_two = 14.50410886.

ii) Results obtained after running PCA without centering the dataset :

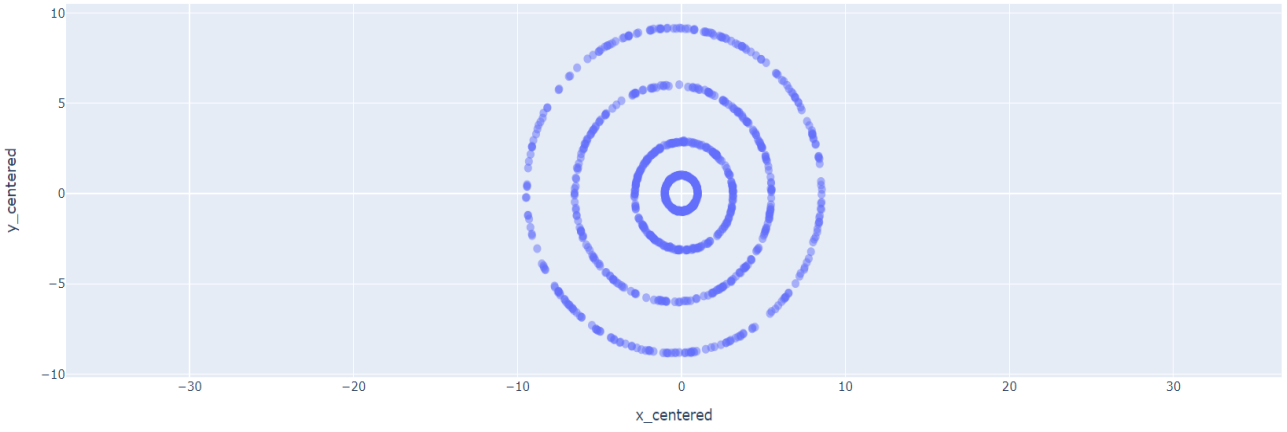
```
Covariance Matrix =
 [[14.7809367   0.80966871]
 [ 0.80966871 16.87223563]]
Eigenvalues :
 [14.50410886 17.14906347]
Eigenvectors :
 [[-0.9462227  0.323516 ]
 [ 0.323516   0.9462227]]
```

I obtained the same eigenvalues(lambda_one, lambda_two) and eigenvectors(w_1, w_2). The mean of the original datapoints is [0 0] that means our original data points are already centered. So, after centering we get the same data points. As a result PCA in this special case gives the same result. Since our original data points are centered, in this case centering is not necessary. Please refer to the below plots -

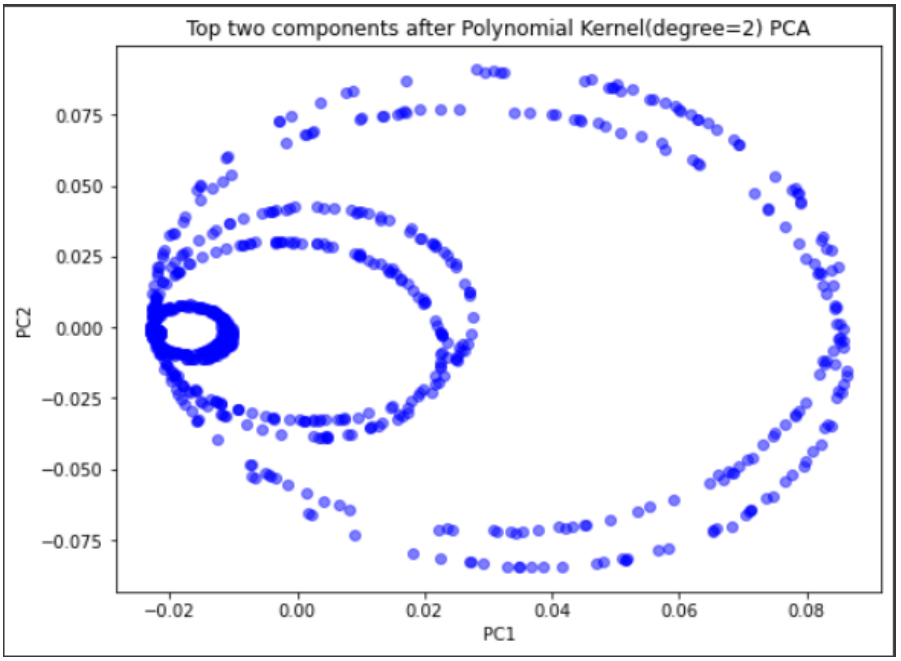
Plot of the data points without centering :



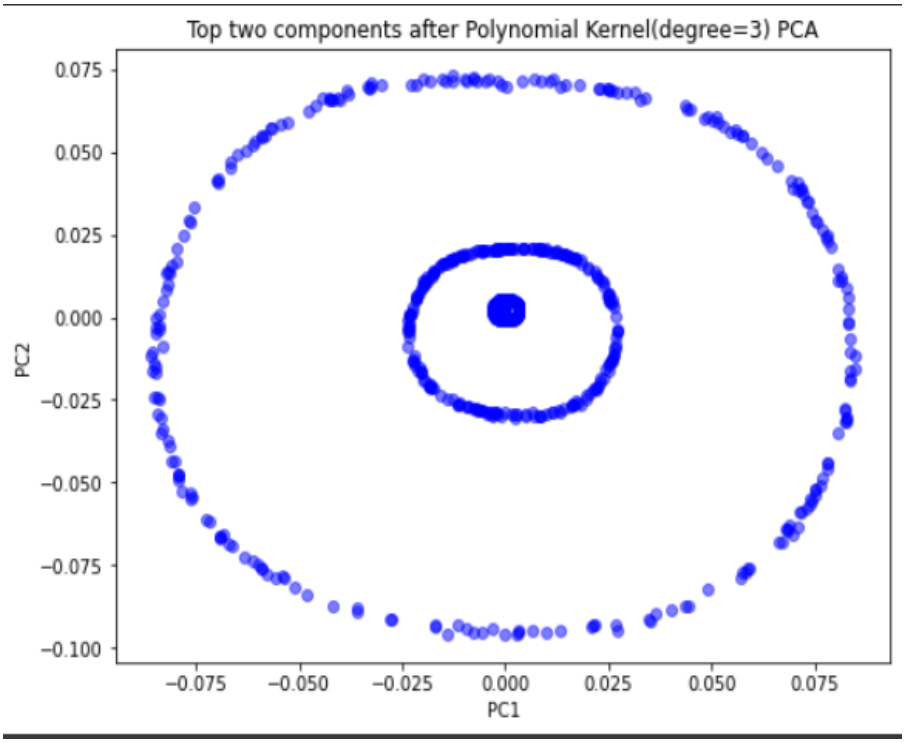
Plot of the data points after centering :



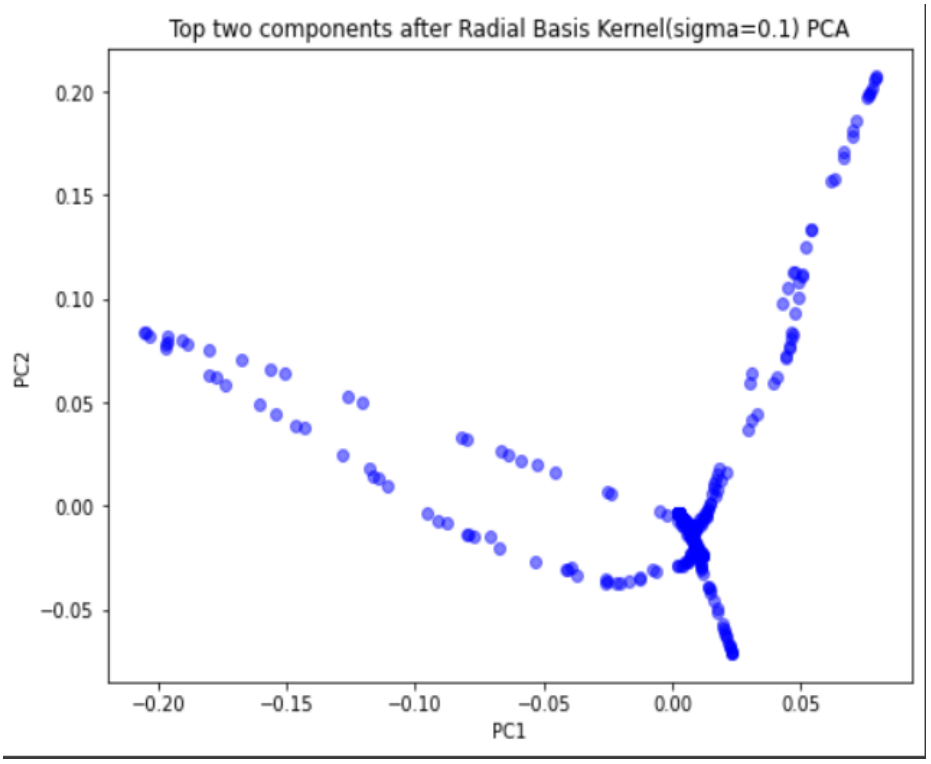
iii) Kernel PCA - Polynomial Kernel (degree = 2) :



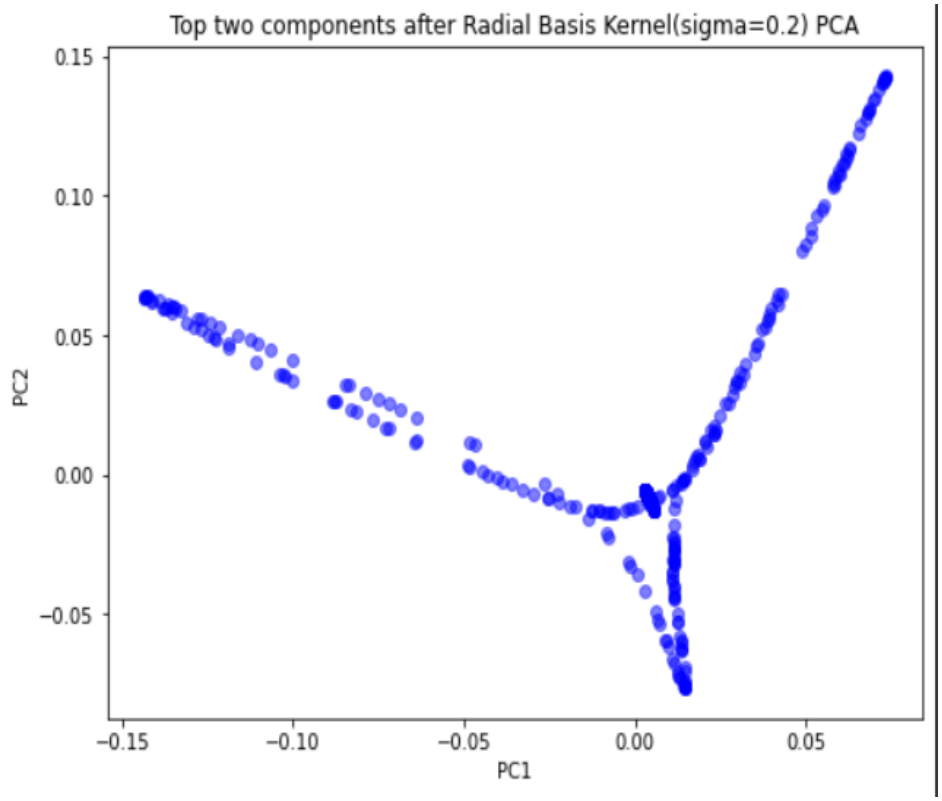
Kernel PCA - Polynomial Kernel (degree = 3) :



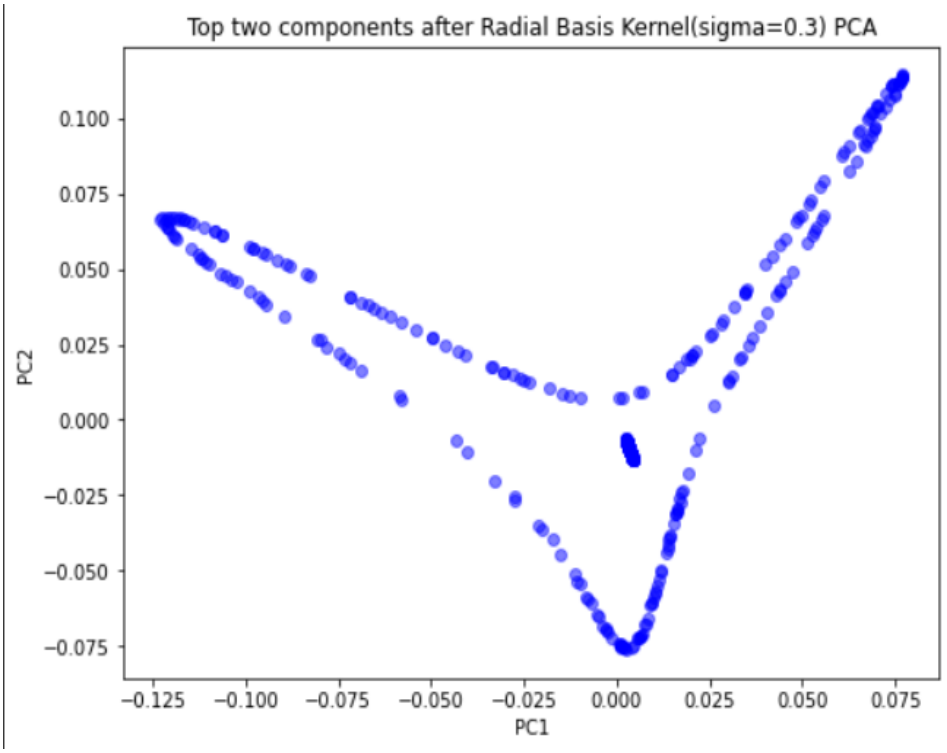
Kernel PCA : Radial Basis Function (sigma = 0.1) :



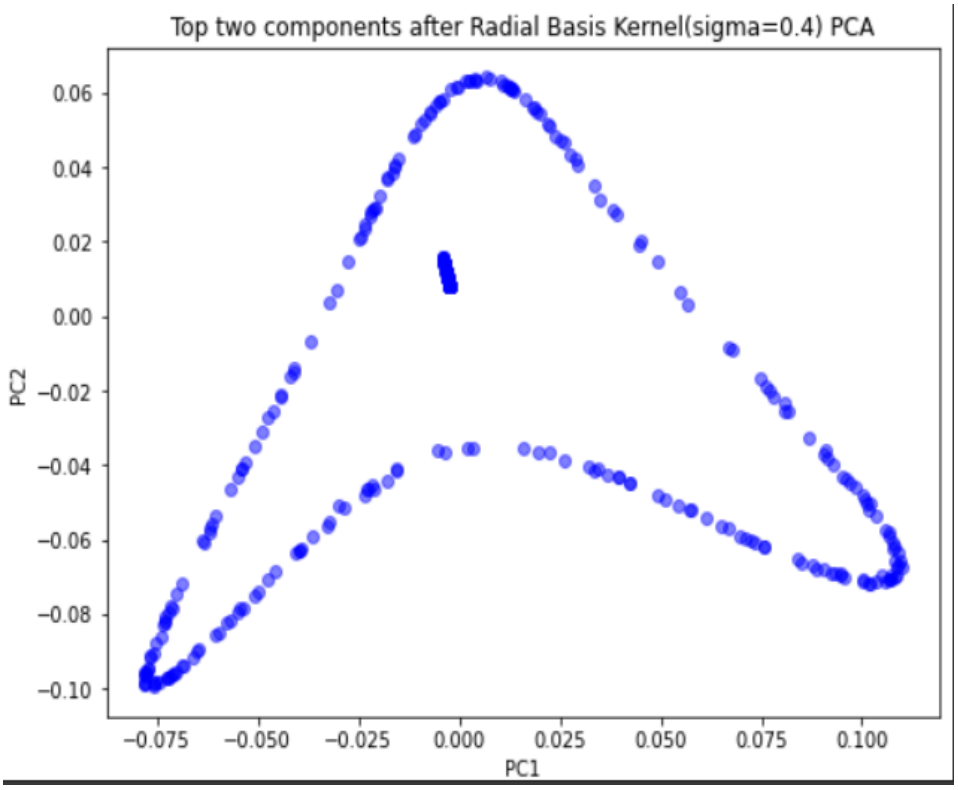
Kernel PCA : Radial Basis Function (sigma = 0.2) :



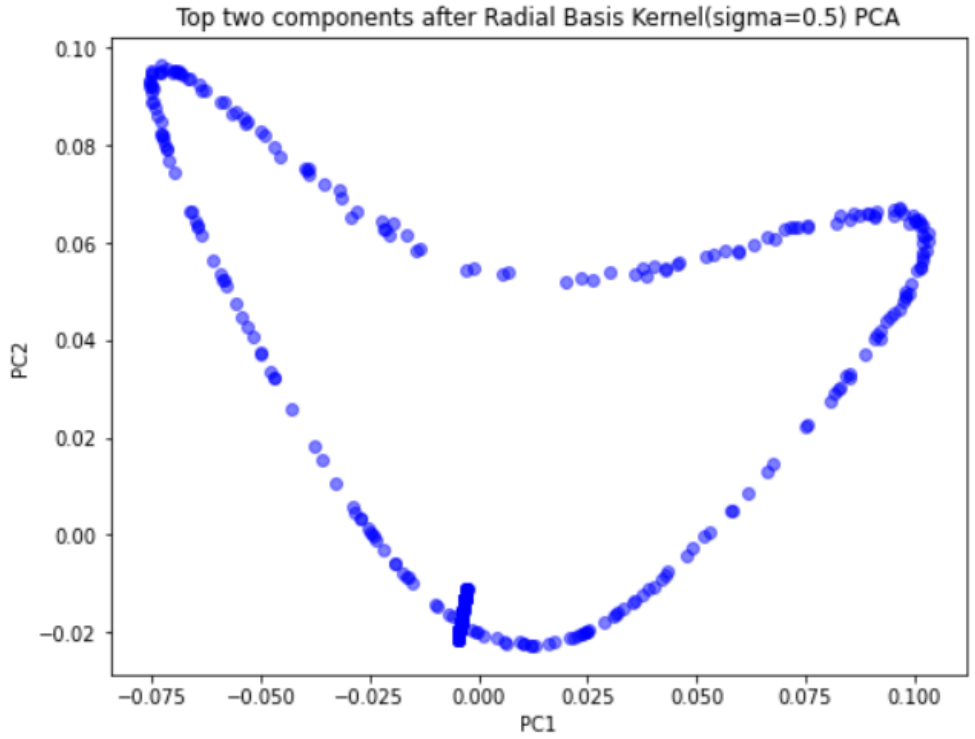
Kernel PCA : Radial Basis Function (sigma = 0.3) :



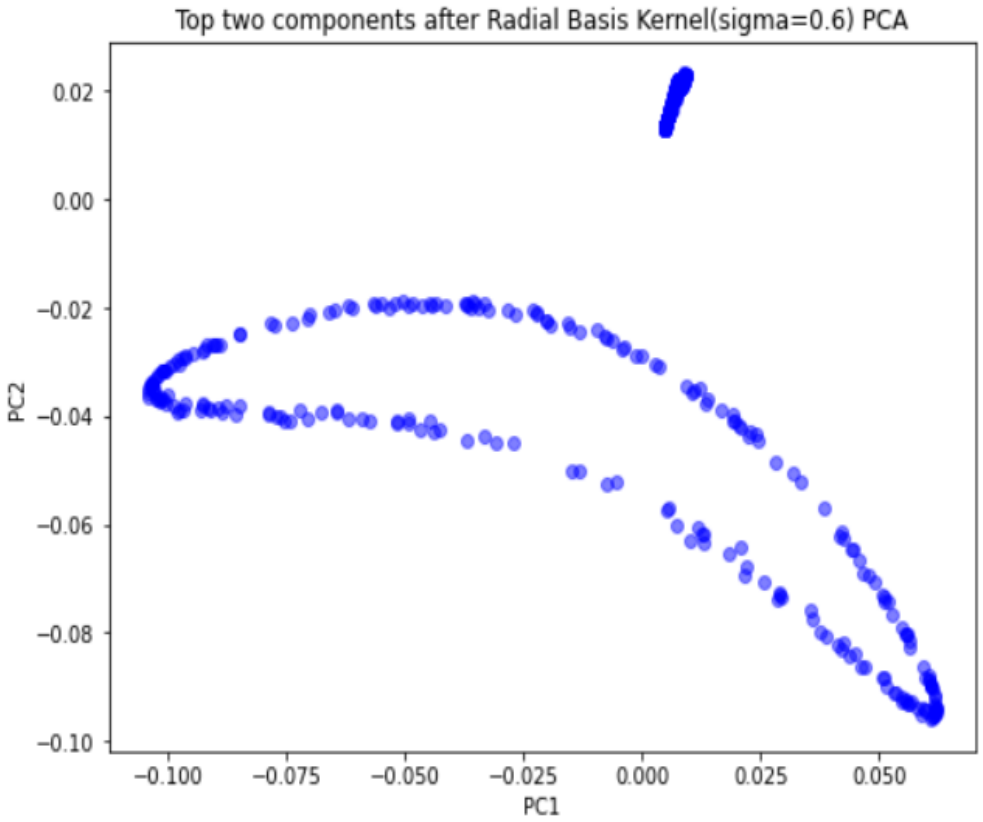
Kernel PCA : Radial Basis Function (sigma = 0.4) :



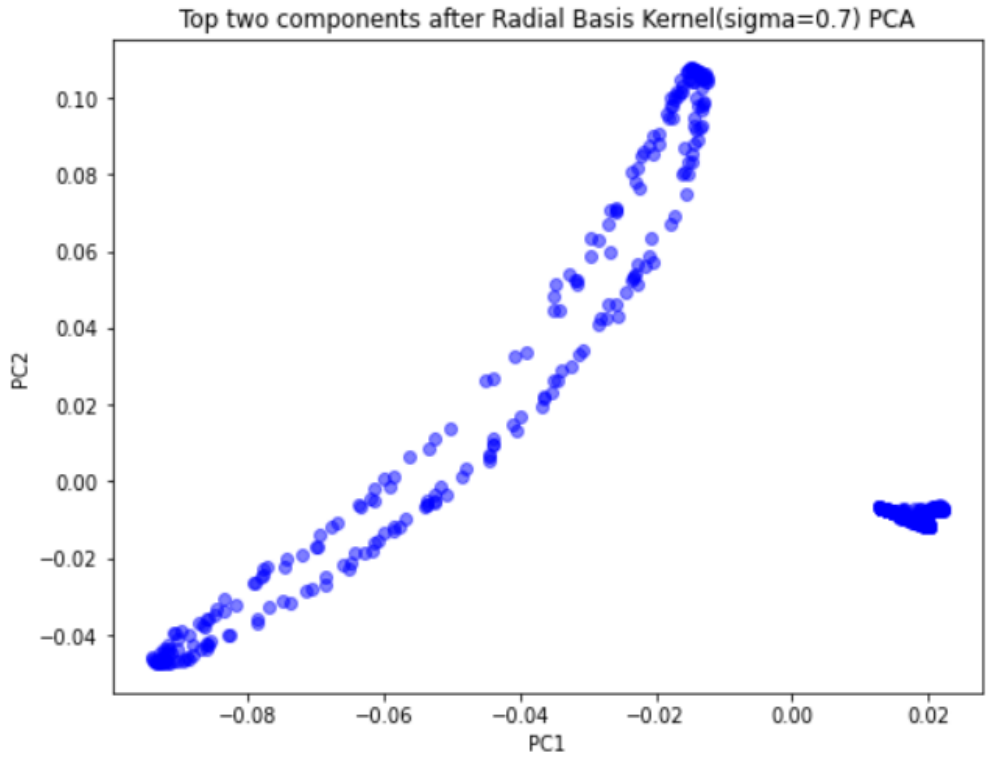
Kernel PCA : Radial Basis Function (sigma = 0.5) :



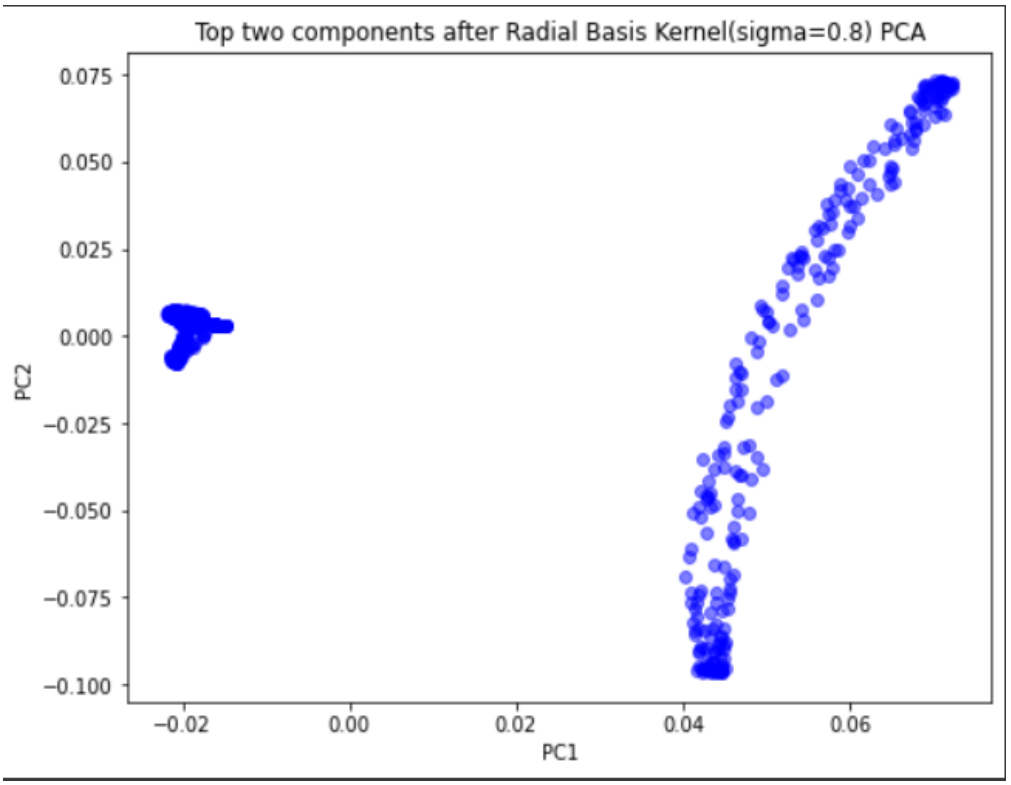
Kernel PCA : Radial Basis Function (sigma = 0.6) :



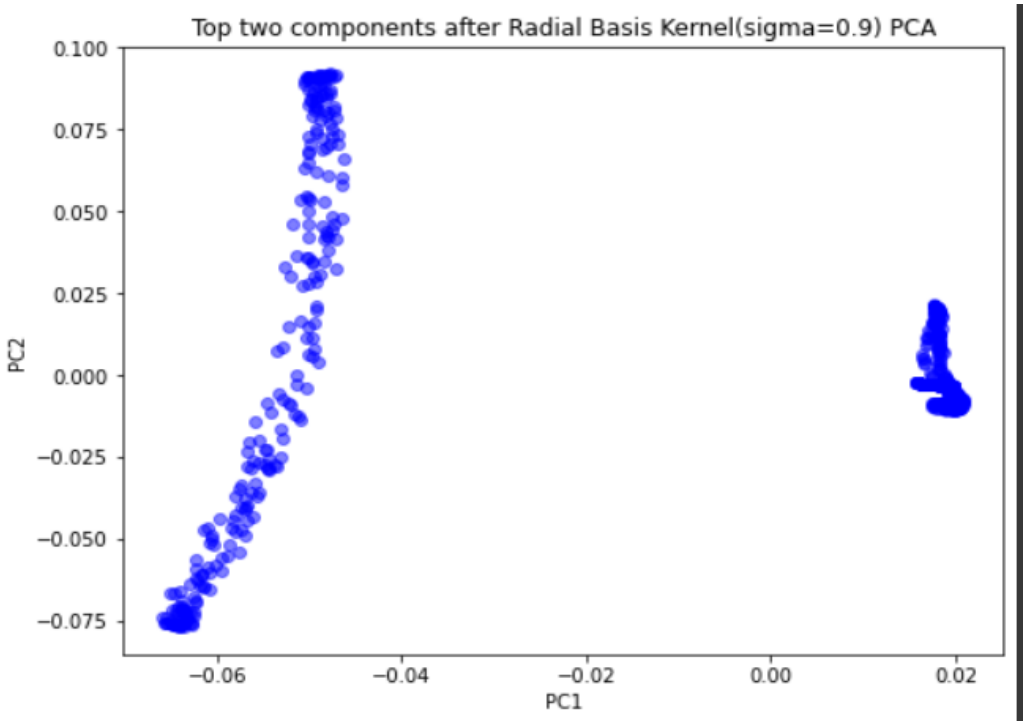
Kernel PCA : Radial Basis Function (sigma = 0.7) :



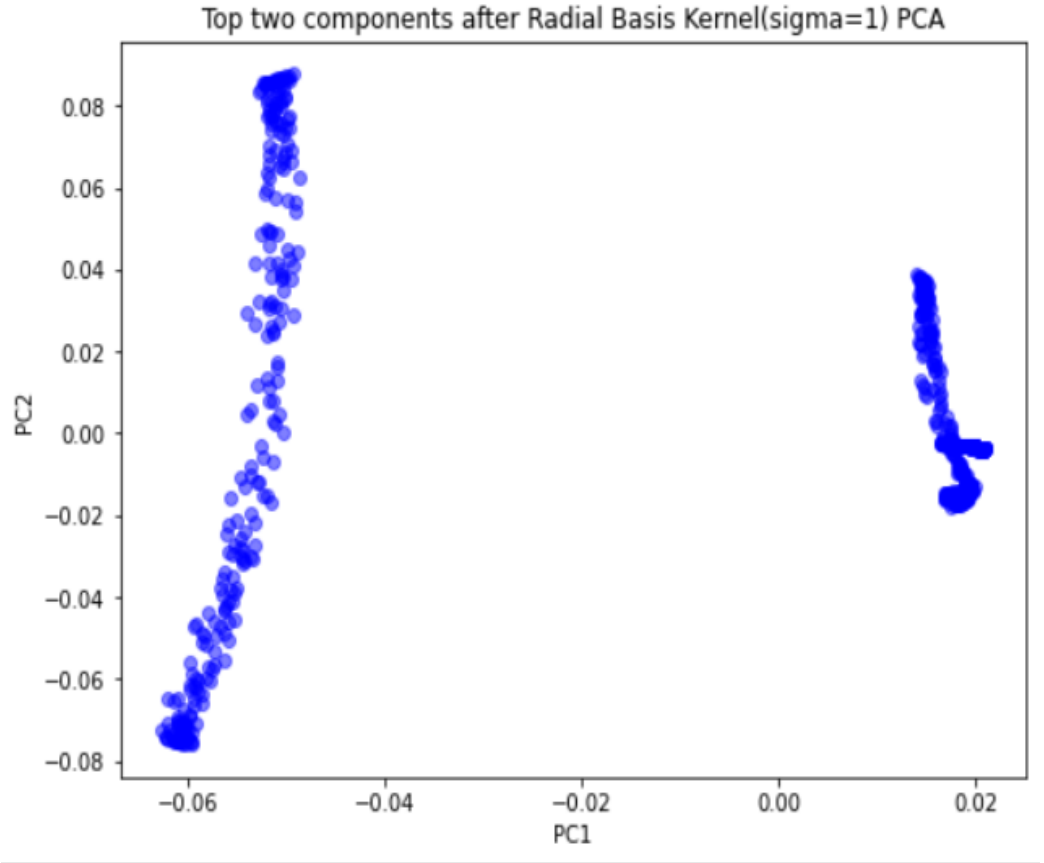
Kernel PCA : Radial Basis Function (sigma = 0.8) :



Kernel PCA : Radial Basis Function (sigma = 0.9) :



Kernel PCA : Radial Basis Function (sigma = 1) :



iv)

I tried two kernels on this dataset - a) Polynomial Kernel and b) RBF Kernel.

For the polynomial kernel, I tried two variants(degree = 2 and degree = 3). The associated plots are present in the previous question. It is clear from the plot(degree=2) that PC1 and PC2 do not produce a subspace which captures the variation in the data well. If we project the data points onto PC1 then the resulting one dimensional representation does not tell us about the degree of separation of the data points.

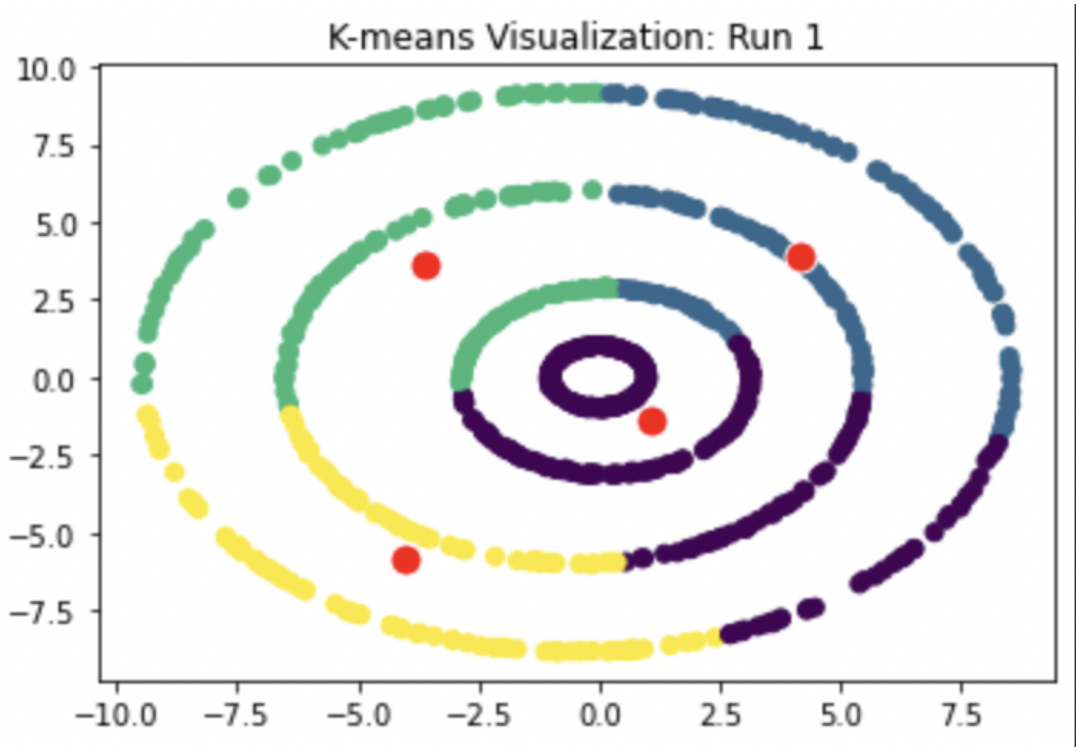
On the other hand when I tried with degree = 3, it is clear from the plot that PC1 did a better job of capturing the degree of separation of the data points. Also the resulting subspace of PC1 and PC2 gives us a better representation as it clearly groups similar points. It groups the inner two concentric circles of the original dataset and magnifies the separation between the outer two circles. So between degree = 2 and degree = 3, my choice would be Polynomial Kernel with degree = 3.

For the RBF kernel I tried the following values of sigma - 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. It is evident from the plots that for sigma = 0.1, the resulting subspace does not group similar data points properly. But as we keep increasing sigma, the representations become better and better gradually. For sigma = 0.3, we start to see two groups properly. Then from sigma = 0.6 onwards the degree of separation becomes very noticeable. For sigma = 0.7 the two groups become linearly separable.

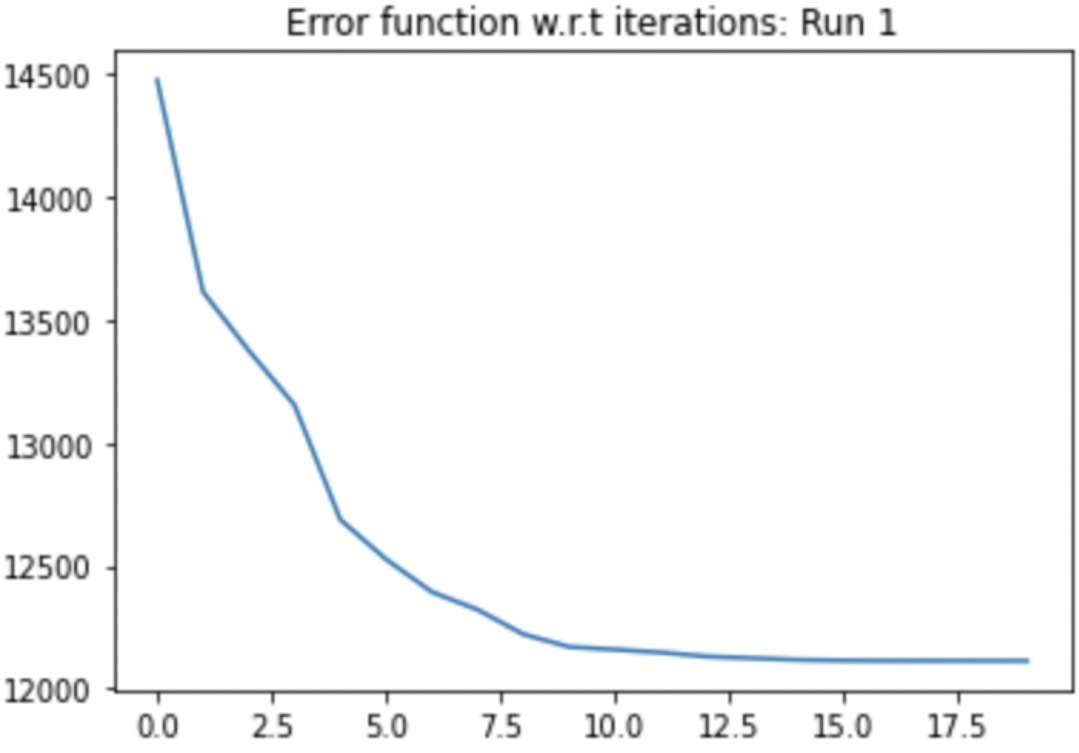
The general question “what is the best kernel for a given problem” depends on the actual downstream task that we intend to solve after obtaining the modified representations. For this particular dataset for sigma = 1, if we project the points on PC1 then the one dimensional subspace obtained looks much better in terms of linear separation. Such a subspace can then be used as input for linear classification models like Support Vector Machines or Naive Bayes Classifiers. So Radial Basis Kernel(sigma = 1) can be a good choice of Kernel for this dataset.

Question : 2

i) Run 1:

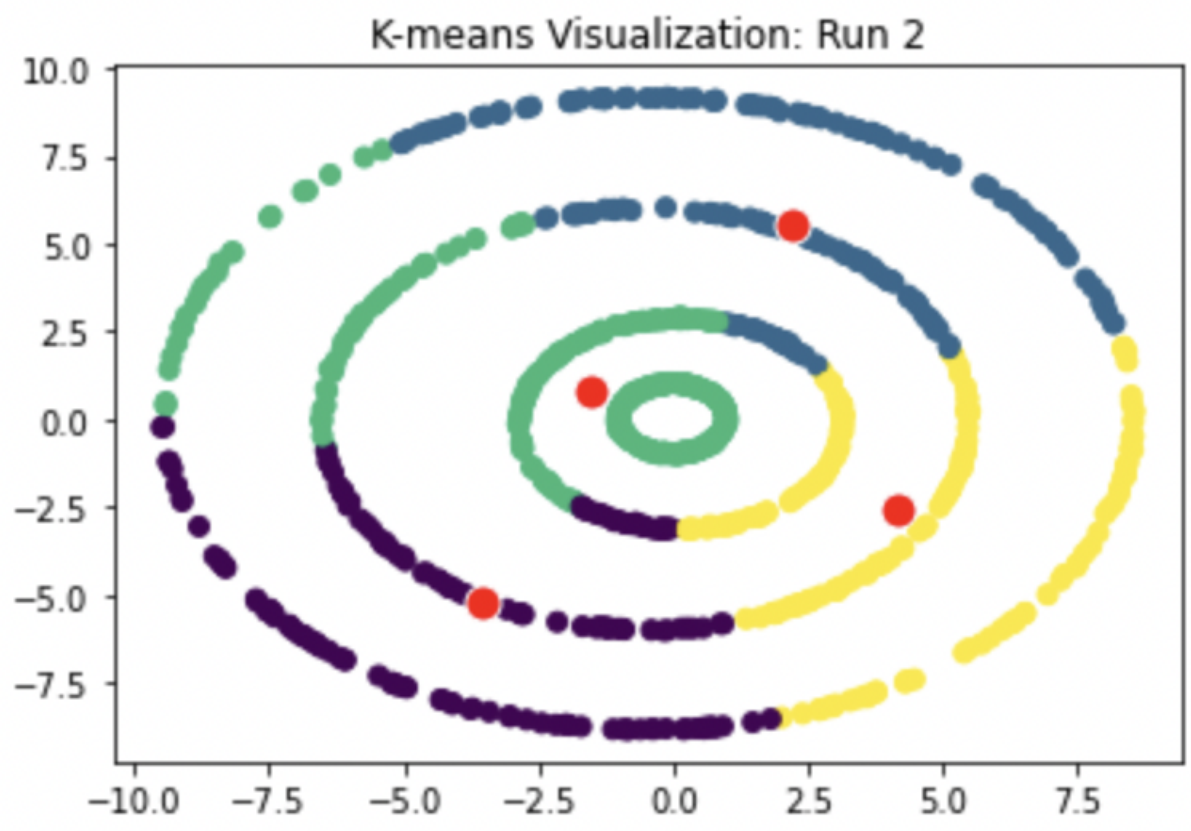


Y axis(vertical) and X axis(horizontal)

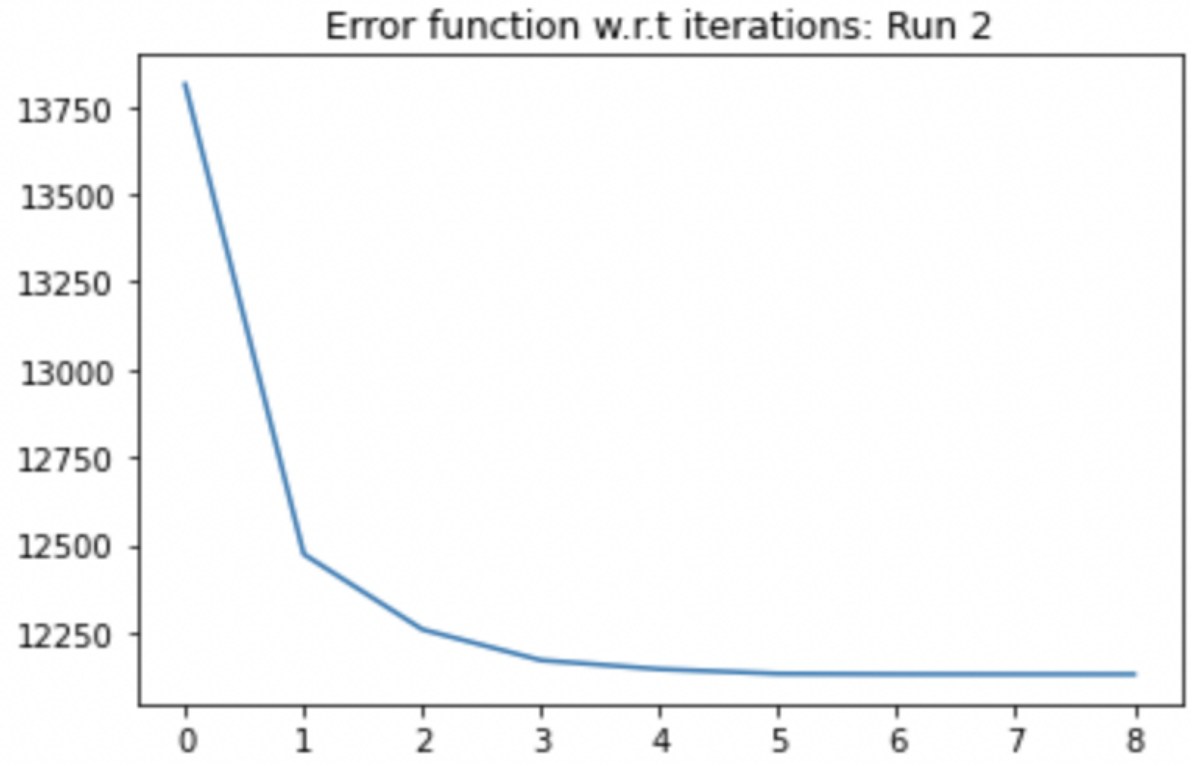


Error (Y axis) vs Iteration Number (X axis)

Run 2:

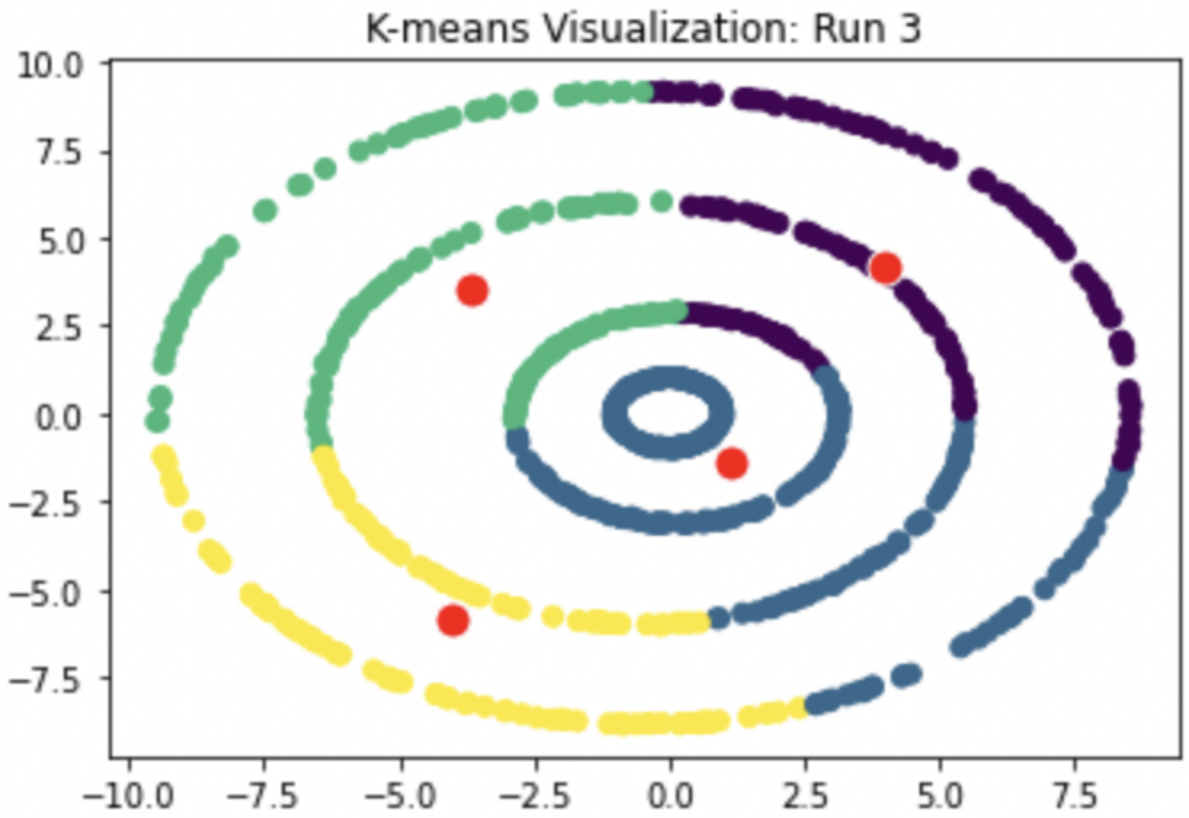


Y axis(vertical) and X axis(horizontal)

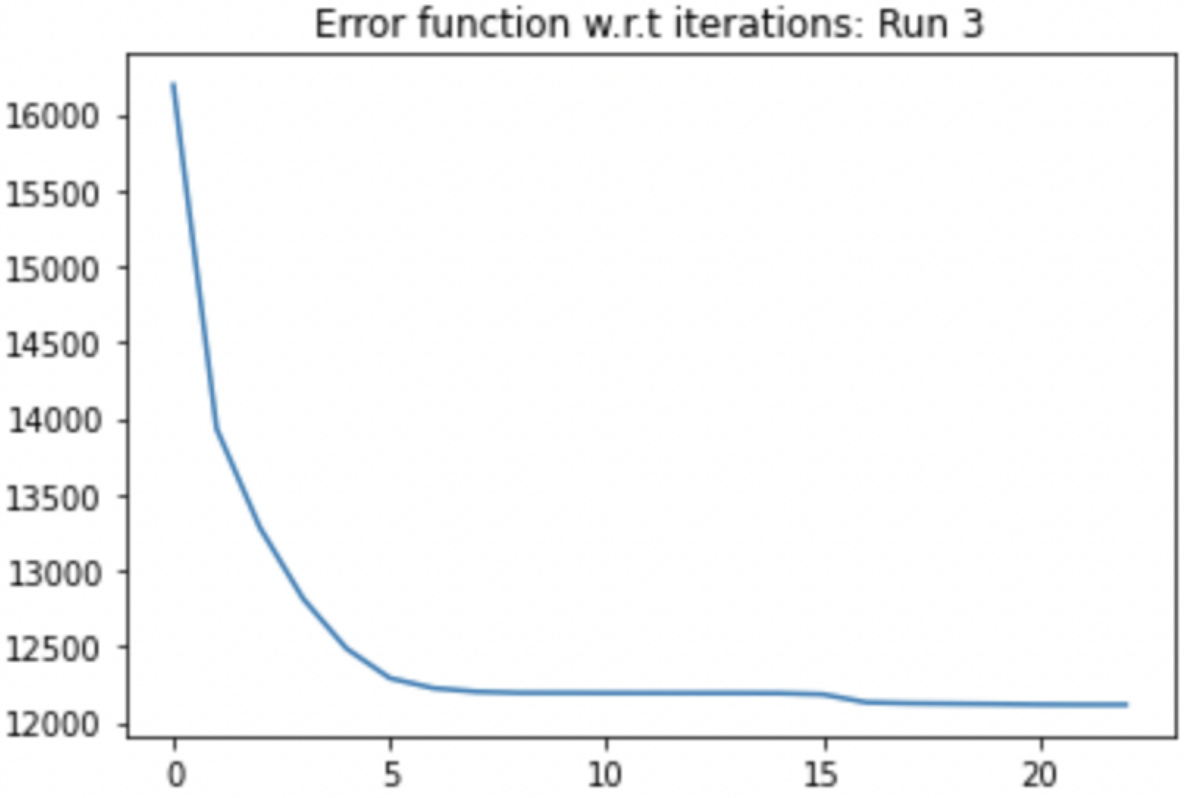


Error (Y axis) vs Iteration Number (X axis)

Run 3:

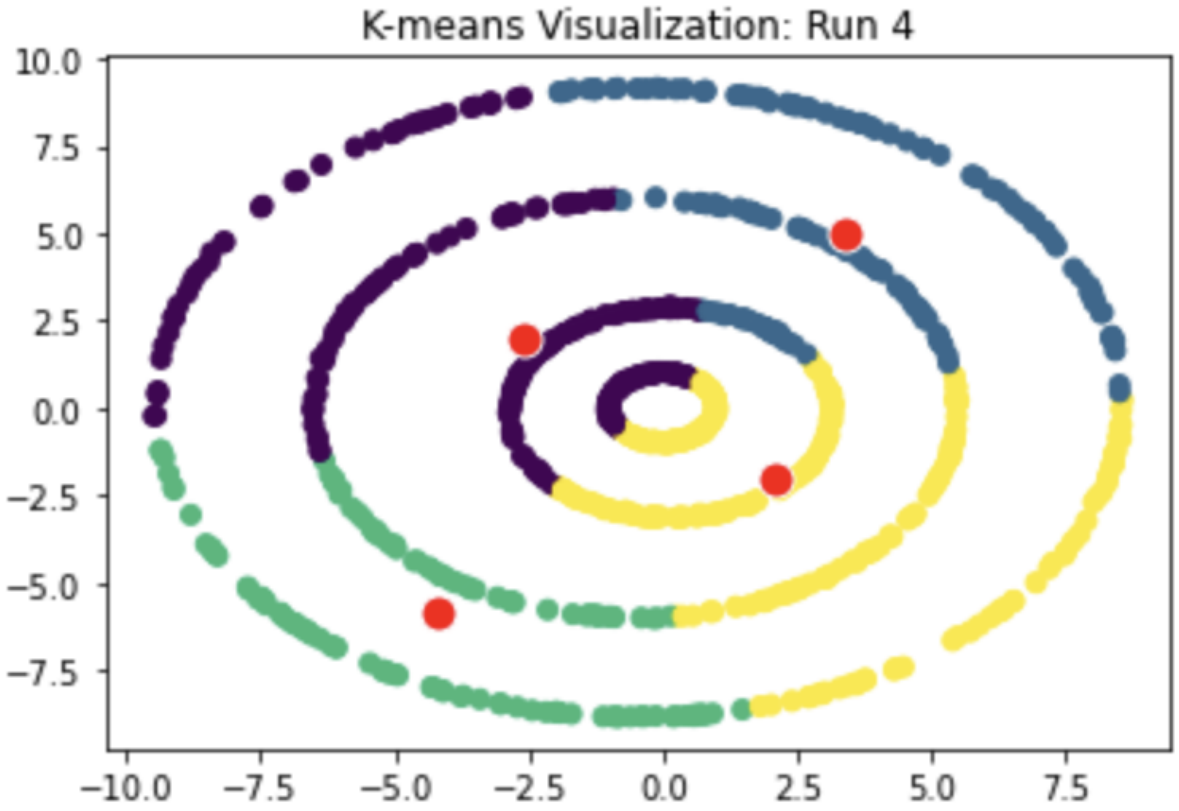


Y axis(vertical) and X axis(horizontal)

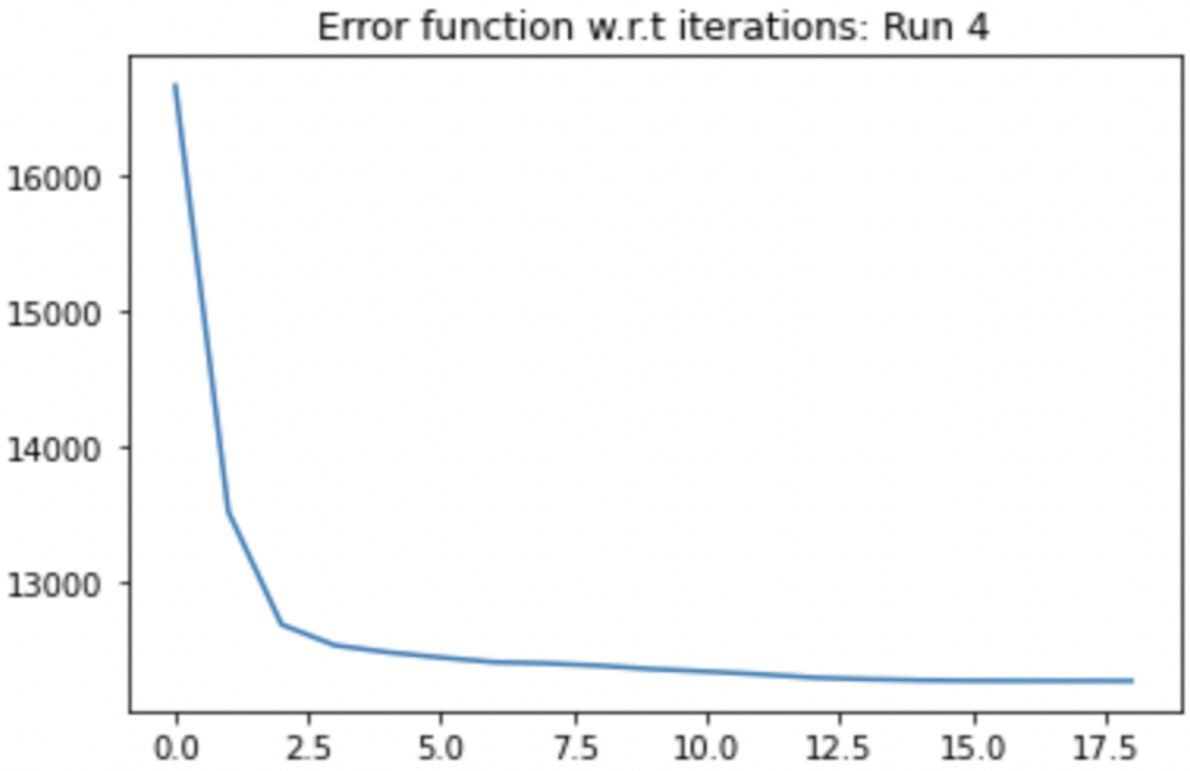


Error (Y axis) vs Iteration Number (X axis)

Run 4:

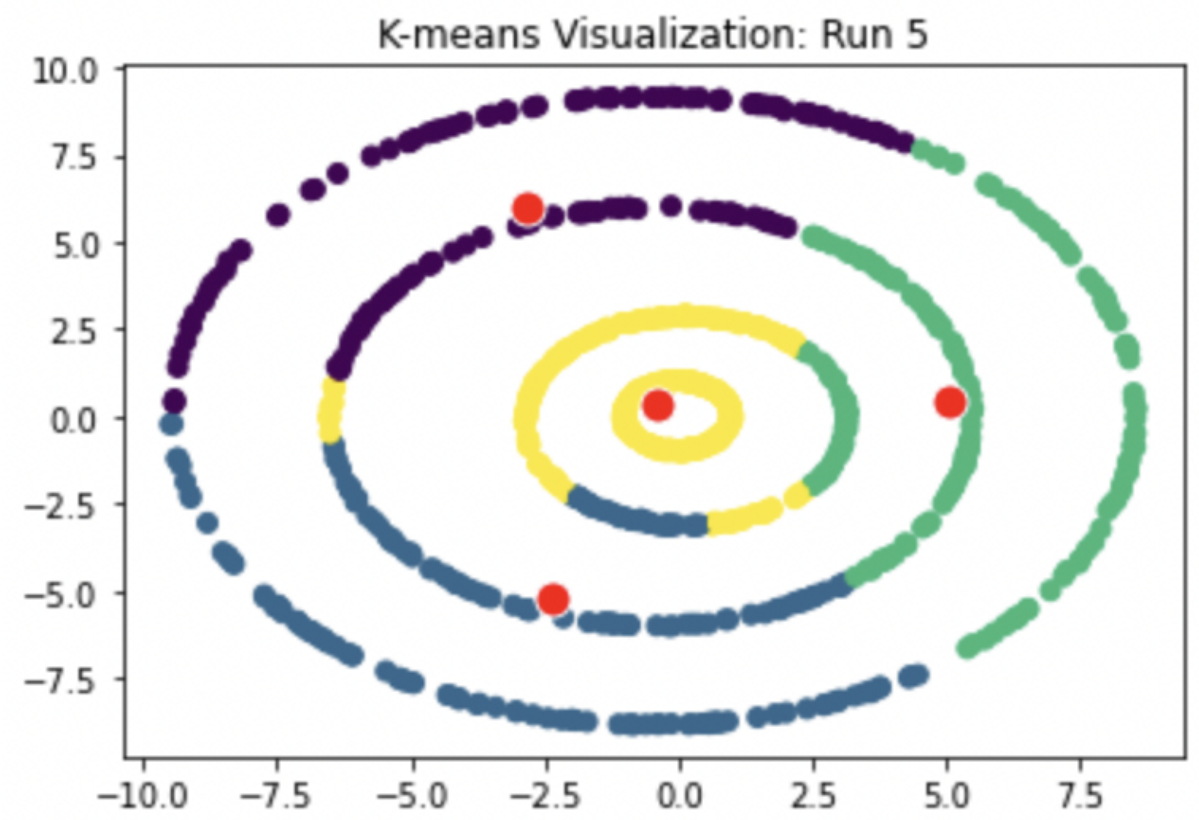


Y axis(vertical) and X axis(horizontal)



Error (Y axis) vs Iteration Number (X axis)

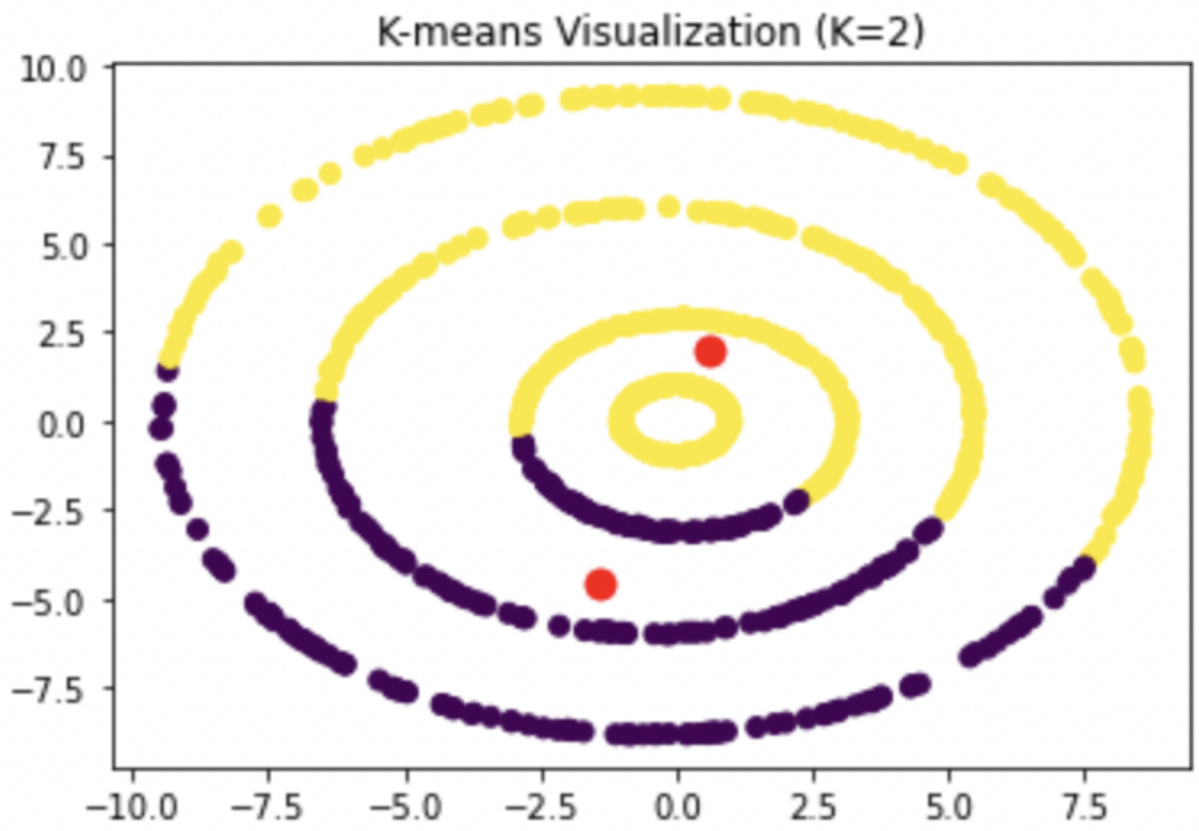
Run 5:



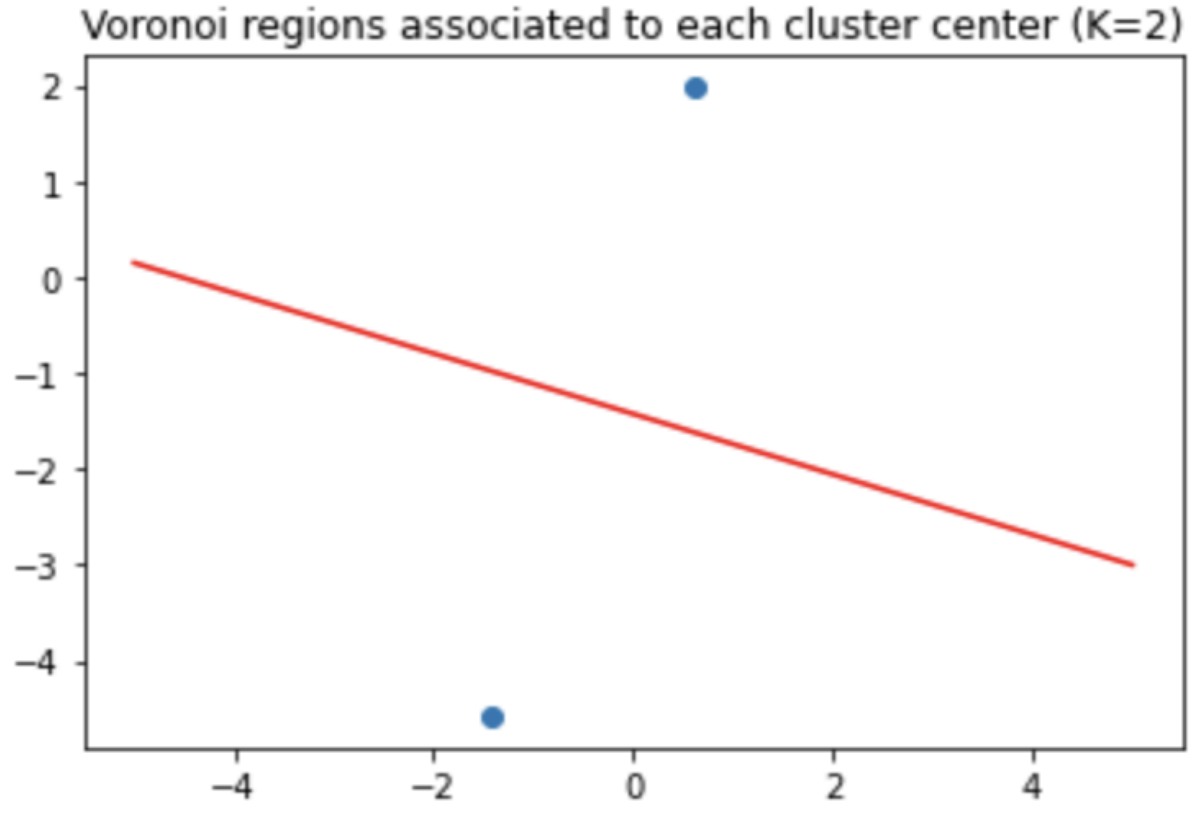
Y axis(vertical) and X axis(horizontal)

Error (Y axis) vs Iteration Number (X axis)

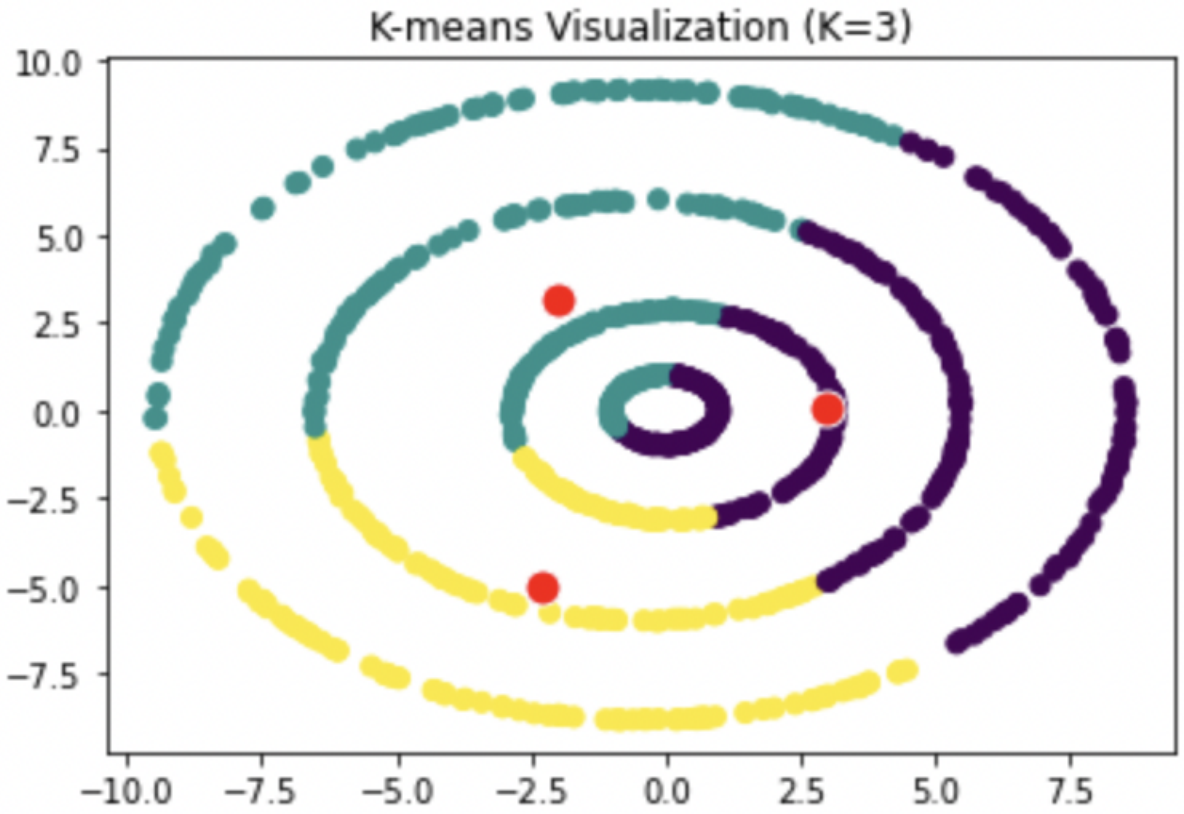
ii)



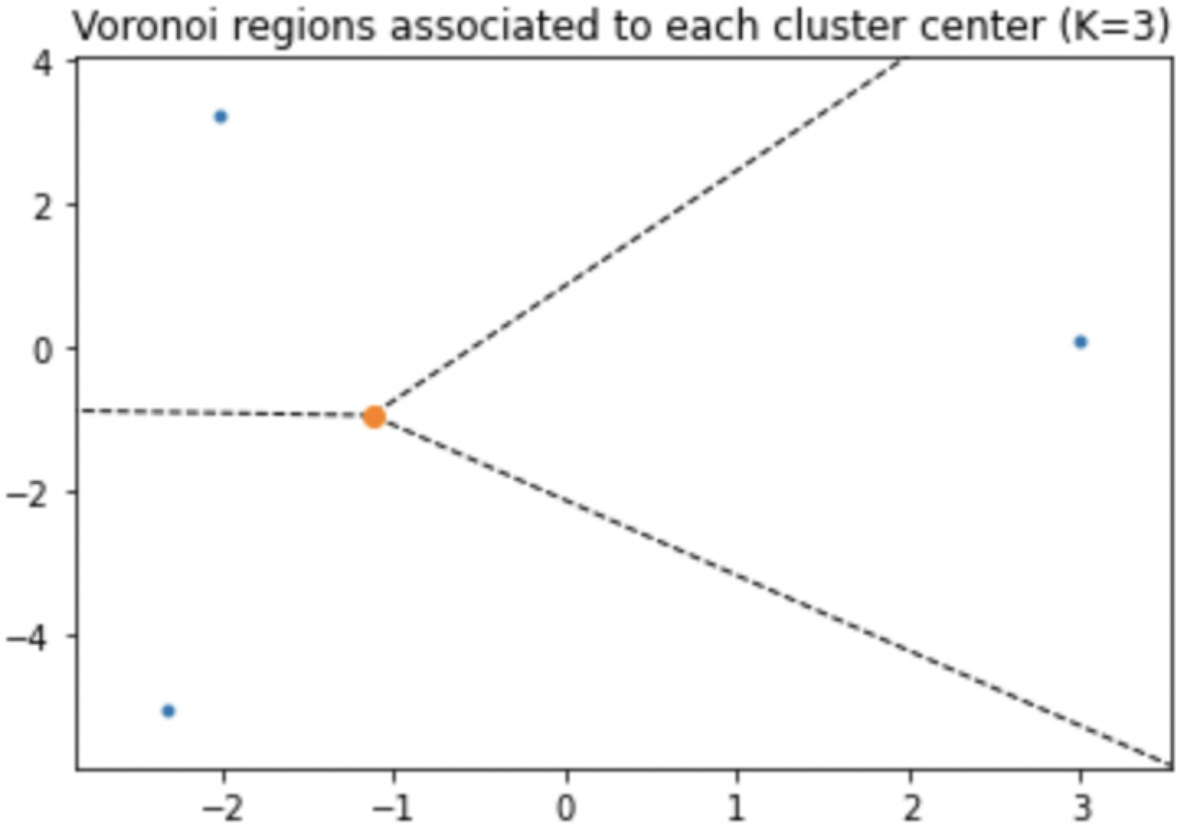
Y axis(vertical) and X axis(horizontal)



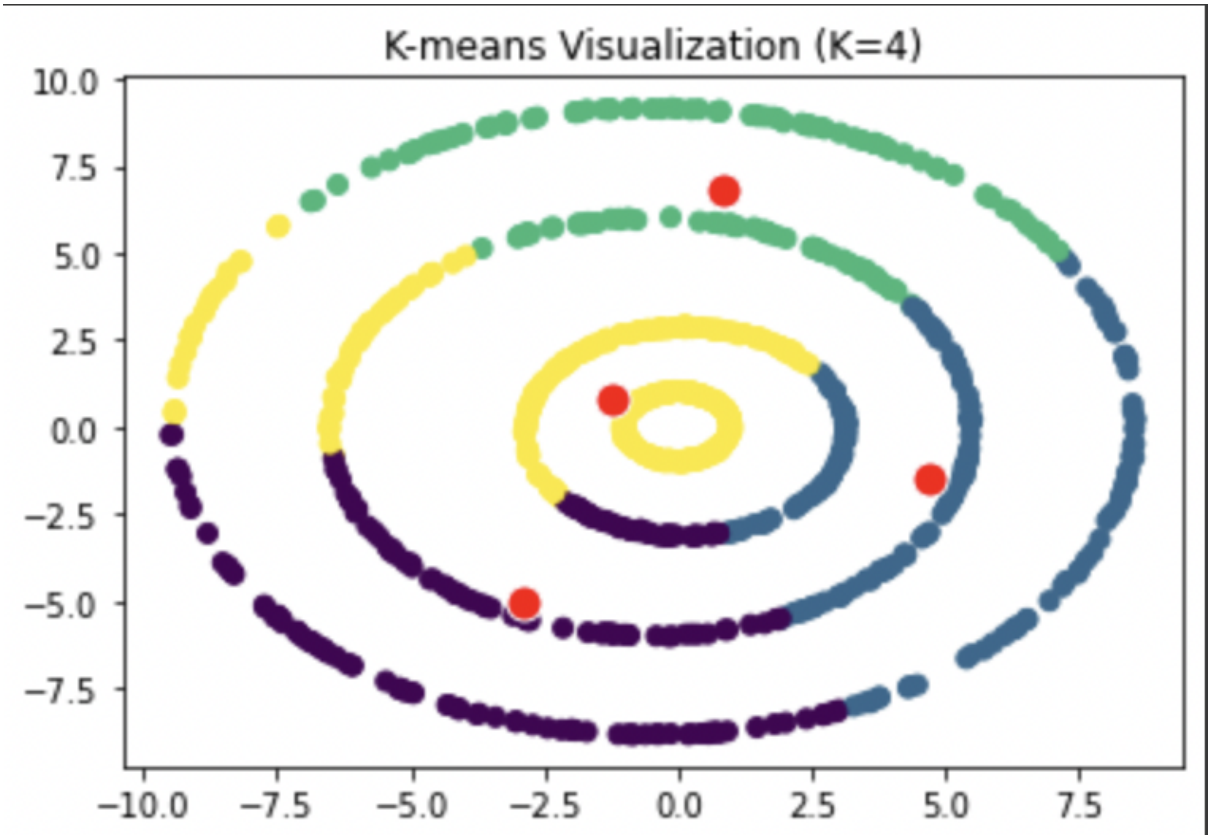
Y axis(vertical) and X axis(horizontal)



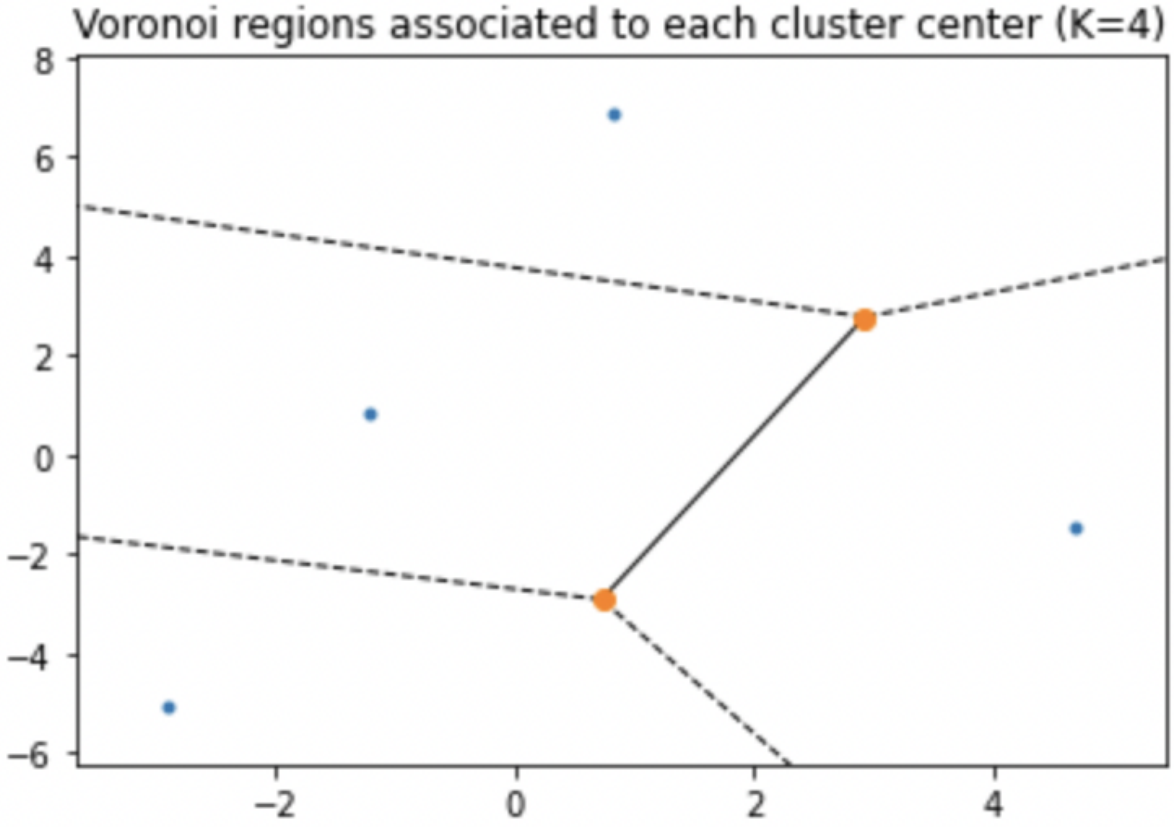
Y axis(vertical) and X axis(horizontal)



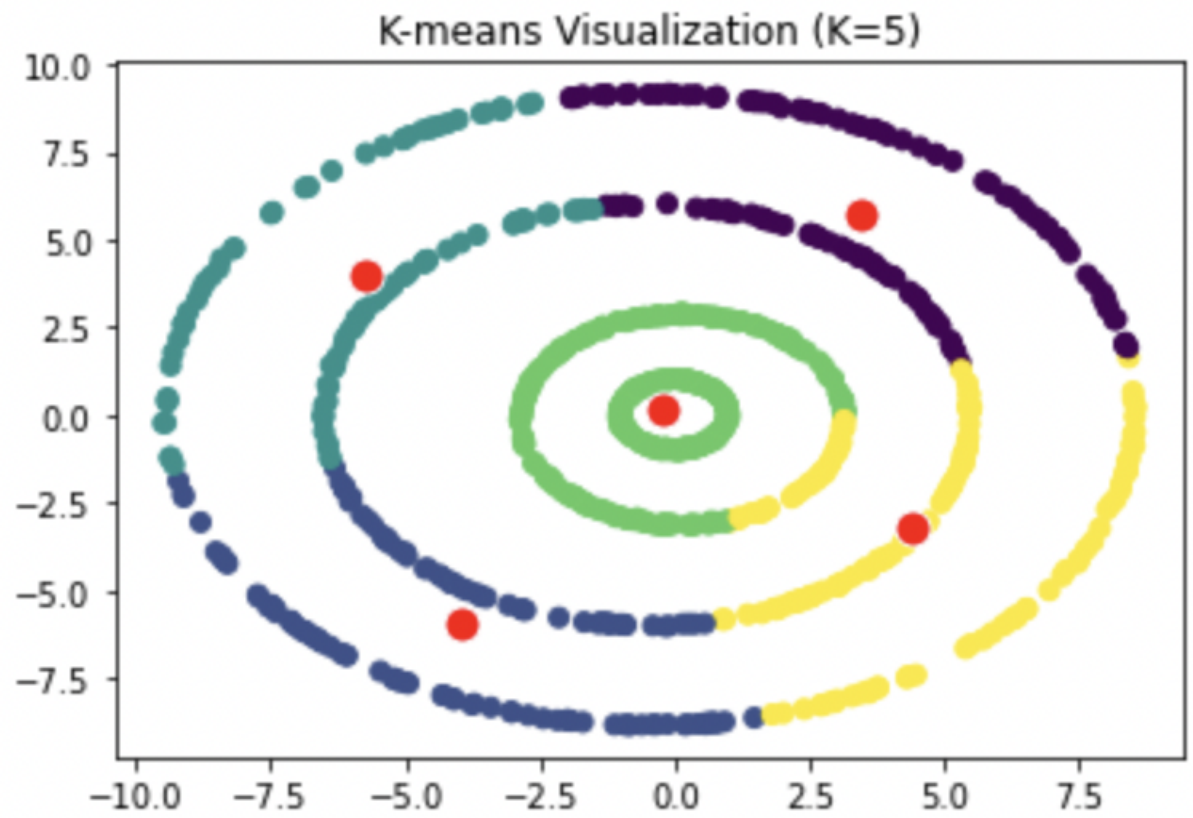
Y axis(vertical) and X axis(horizontal)



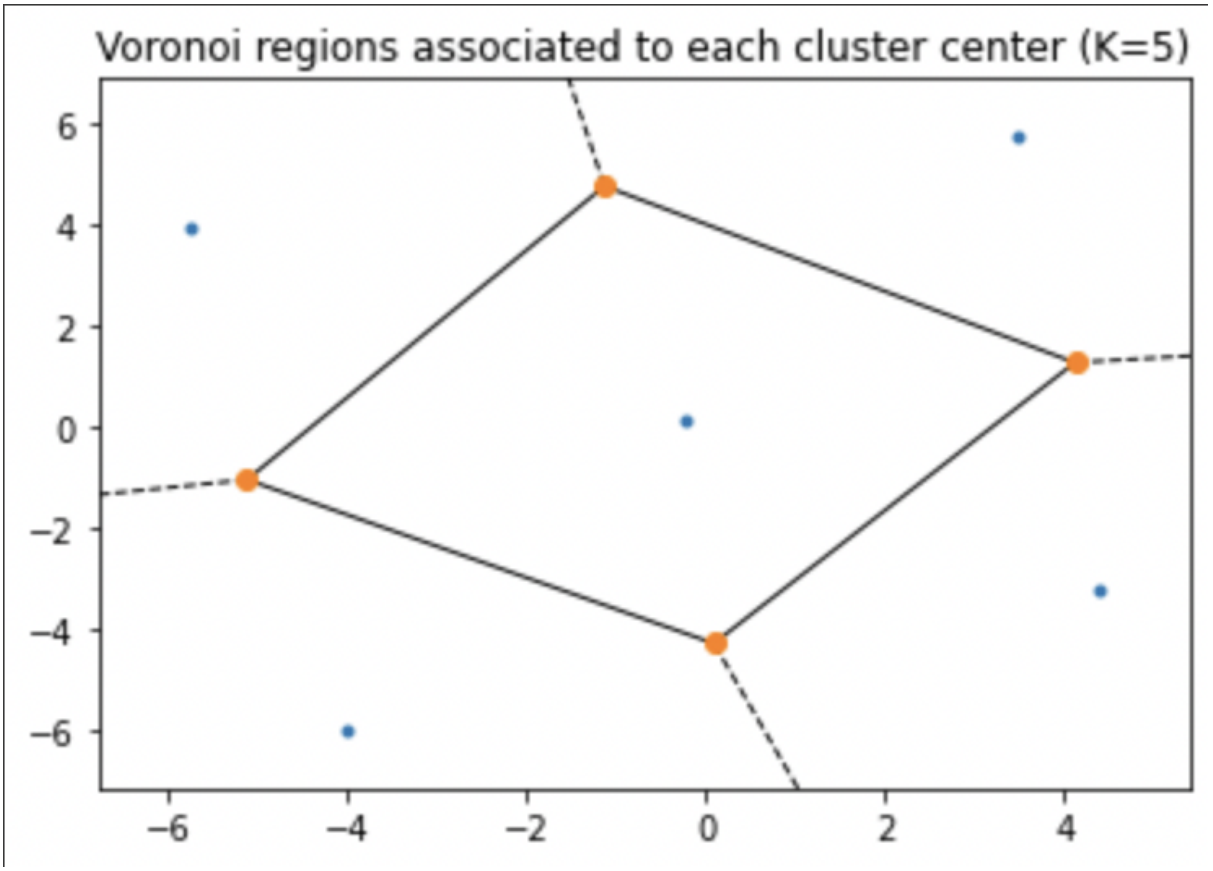
Y axis(vertical) and X axis(horizontal)



Y axis(vertical) and X axis(horizontal)



Y axis(vertical) and X axis(horizontal)

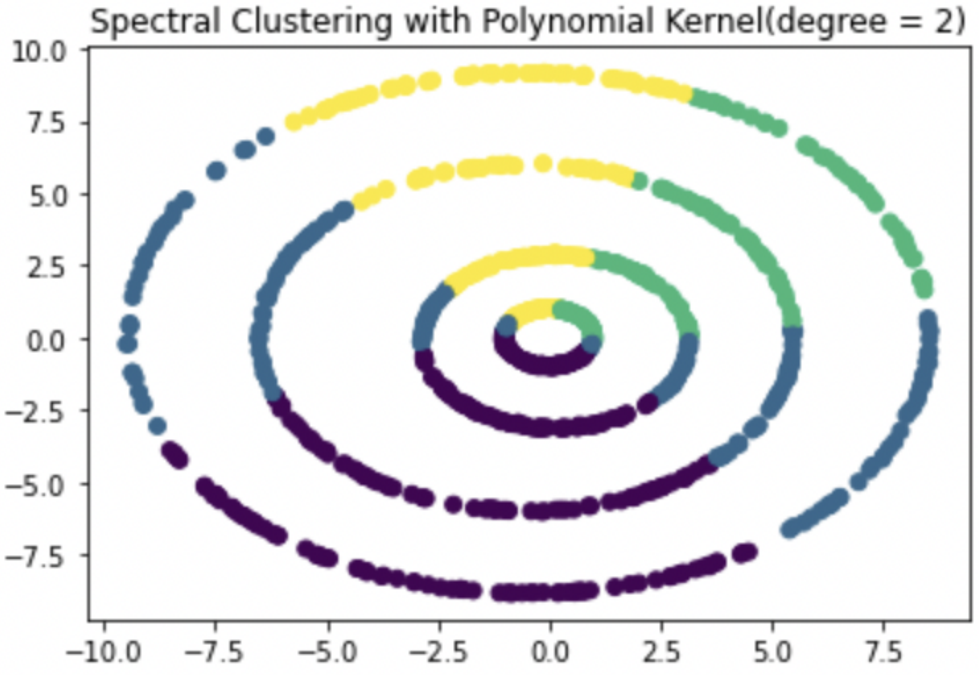


Y axis(vertical) and X axis(horizontal)

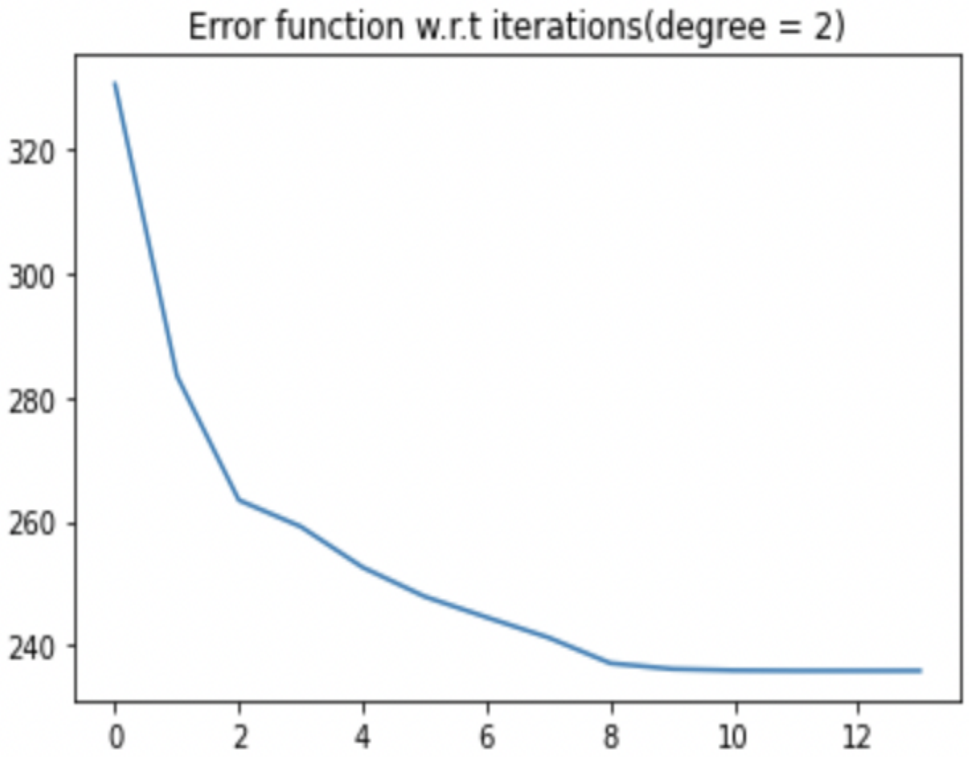
iii)

I ran the spectral clustering algorithm using two Kernels - a) Polynomial Kernel and b) Radial Basis Function Kernel.

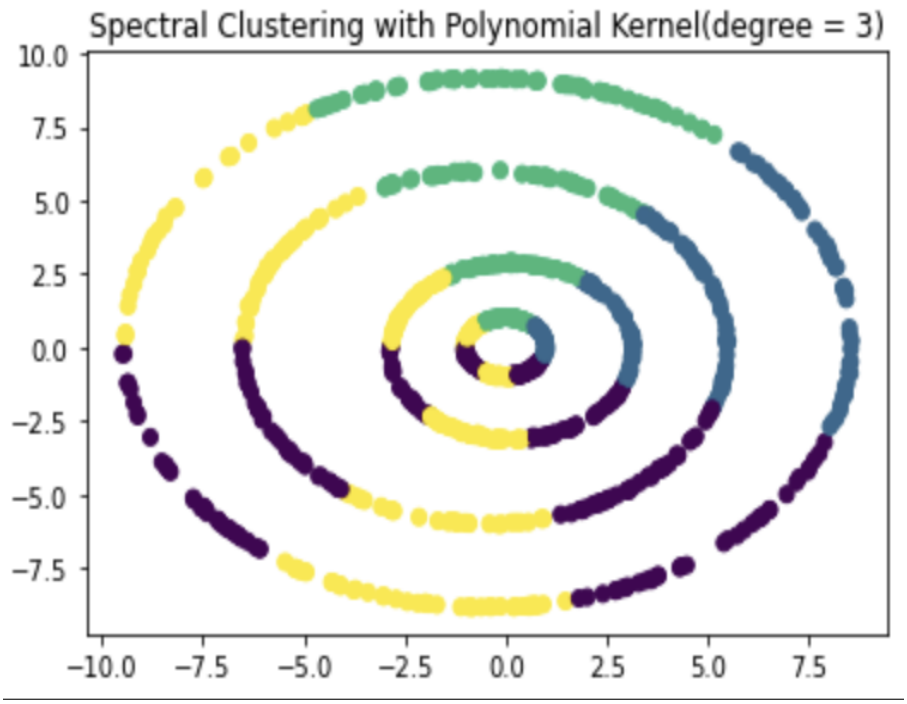
For the Polynomial Kernel, I tried four variants - degree 2, degree 3, degree 4, degree 5. The clusters and the error function plots are given below.



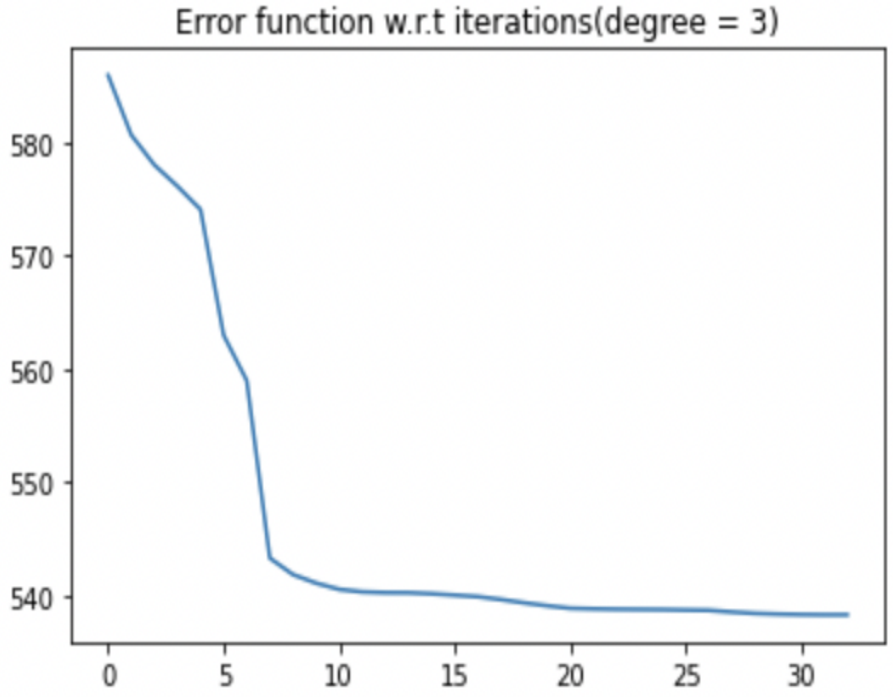
Y axis(vertical) and X axis(horizontal)



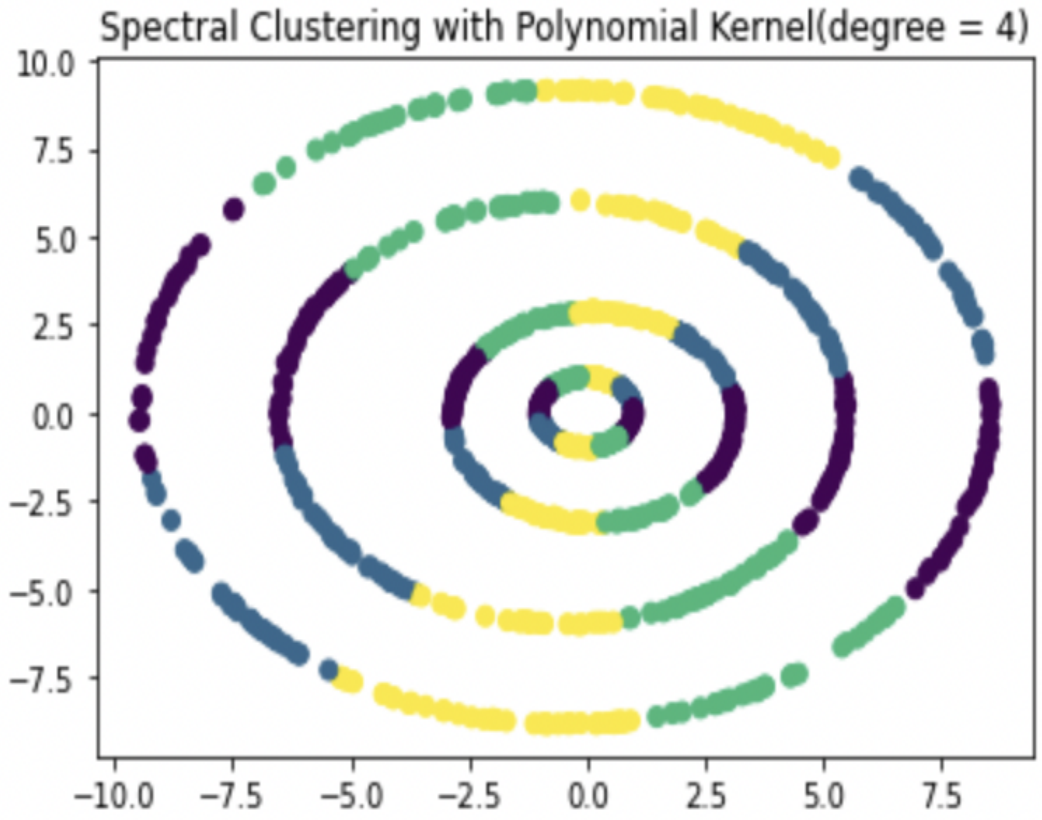
Error (Y axis) vs Iteration Number (X axis)



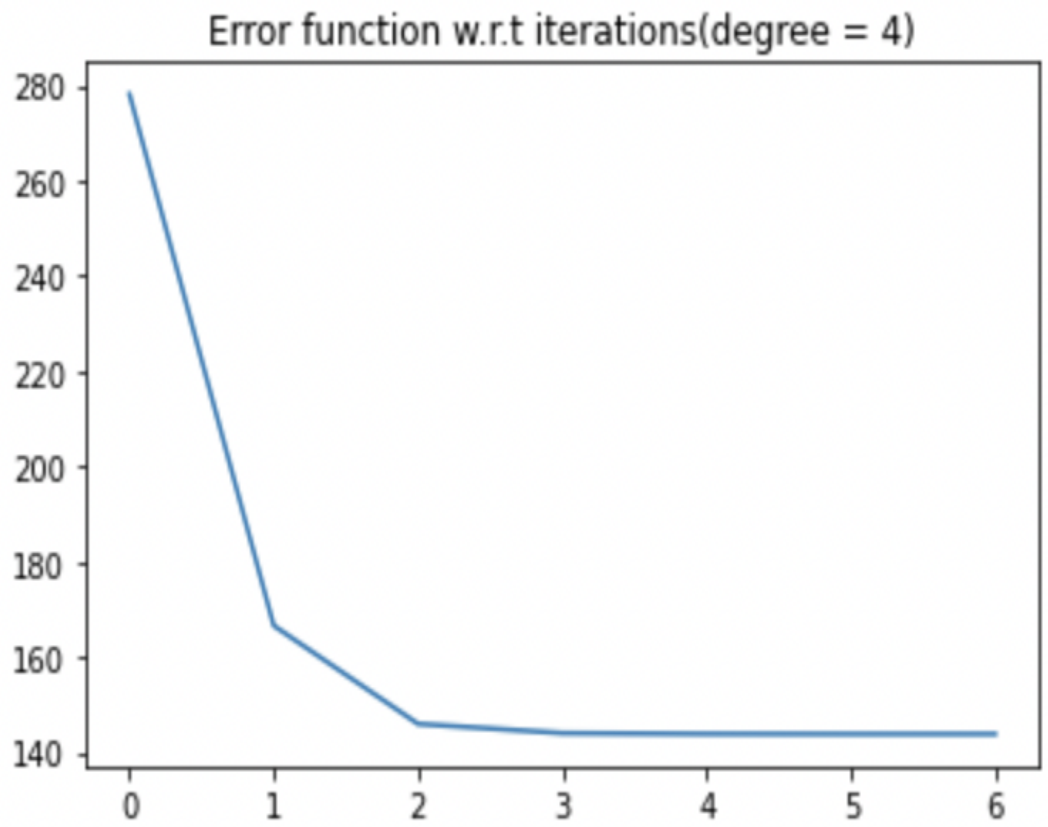
Y axis(vertical) and X axis(horizontal)



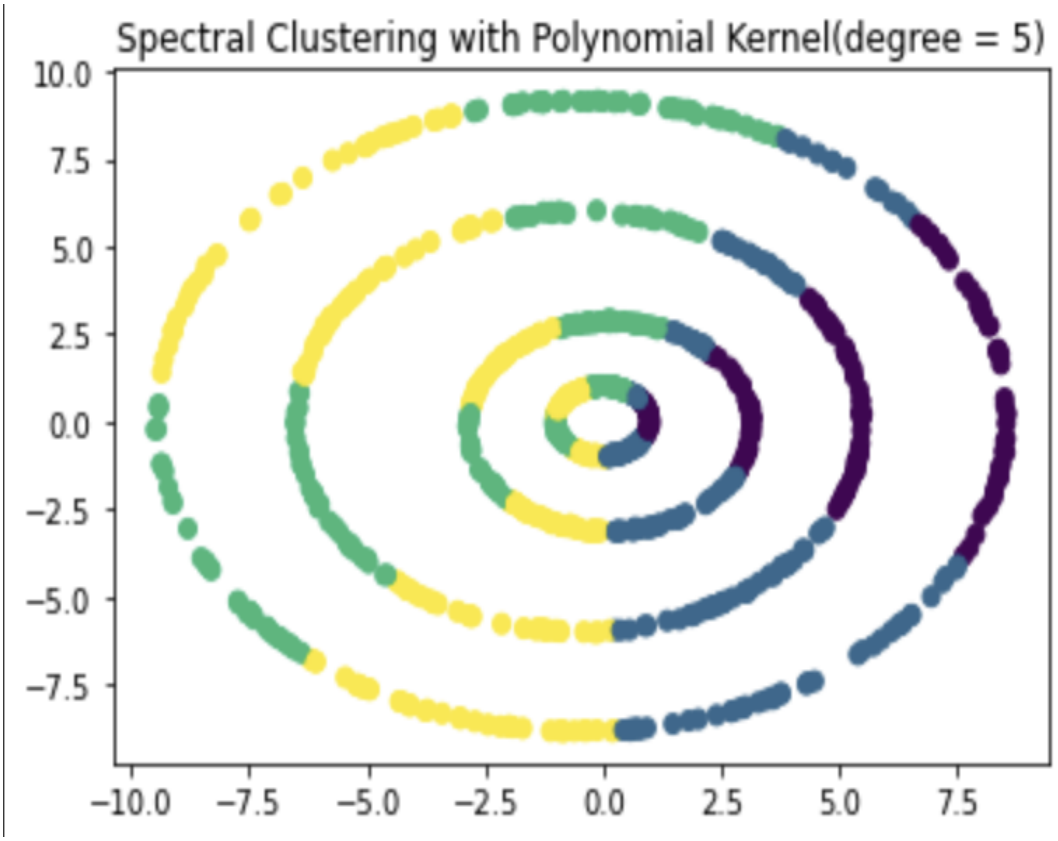
Error (Y axis) vs Iteration Number (X axis)



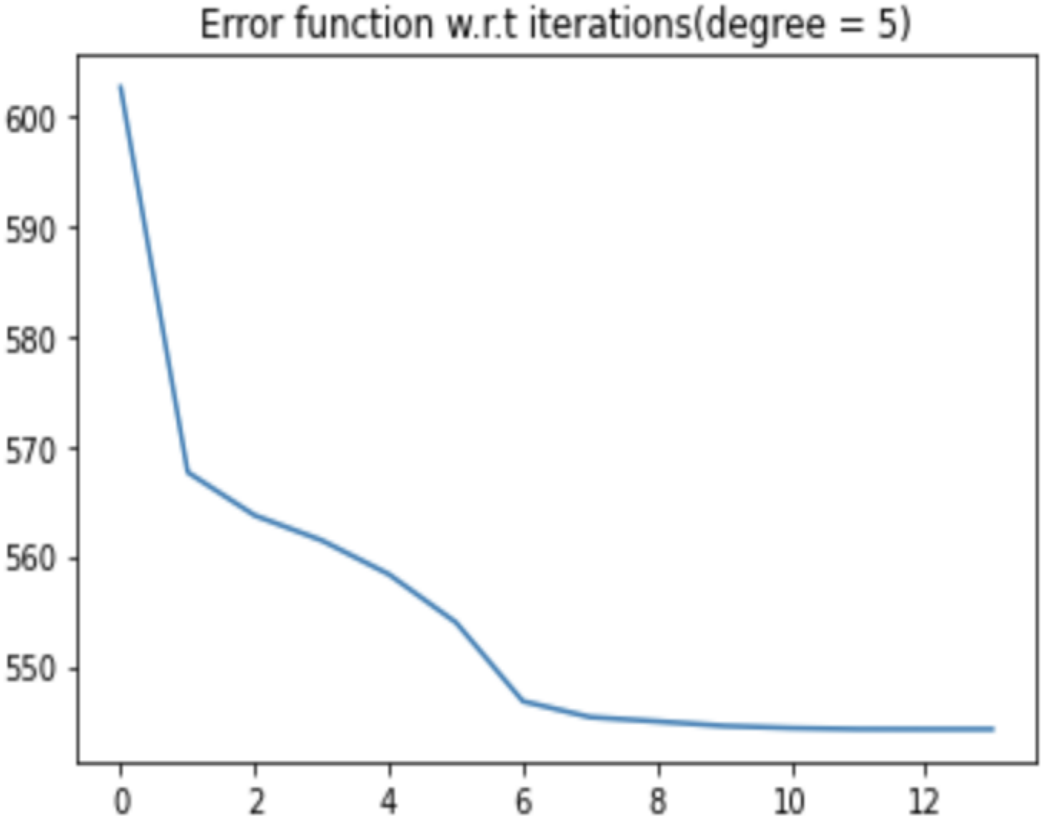
Y axis(vertical) and X axis(horizontal)



Error (Y axis) vs Iteration Number (X axis)



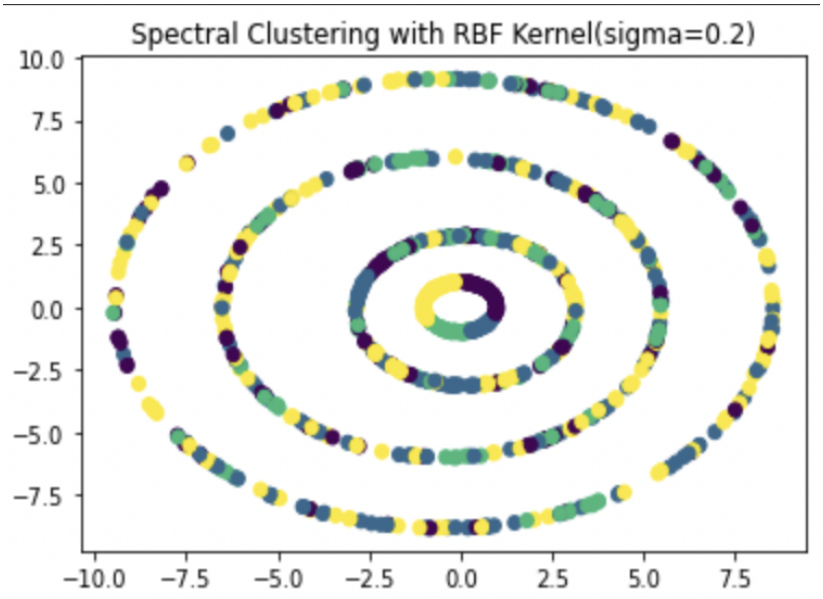
Y axis(vertical) and X axis(horizontal)



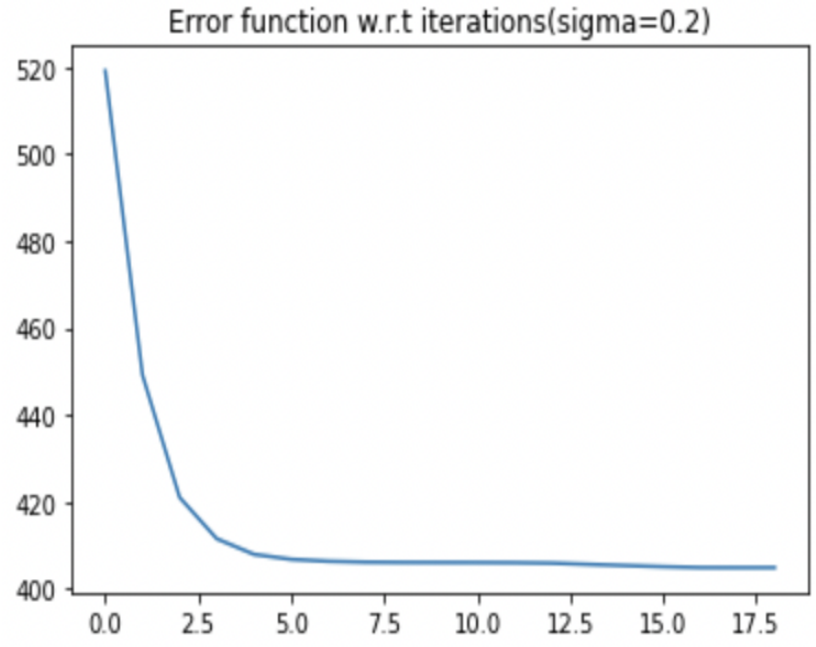
Error (Y axis) vs Iteration Number (X axis)

It is clear from the above error function plots that degree 2 polynomial kernel results in lesser error compared to degree 3 polynomial kernel. And degree 4 polynomial kernel results in lesser error compared to degree 5 polynomial kernel. I tried with degrees 6,7,8 as well. And I observed a general pattern. For this problem, if we pick polynomial kernels whose degrees are even, they perform better than polynomial kernels whose degrees are odd. It is obvious from the above plots that the degree 4 polynomial kernel gives us the minimum error. So among polynomial kernels that would be my choice for this problem.

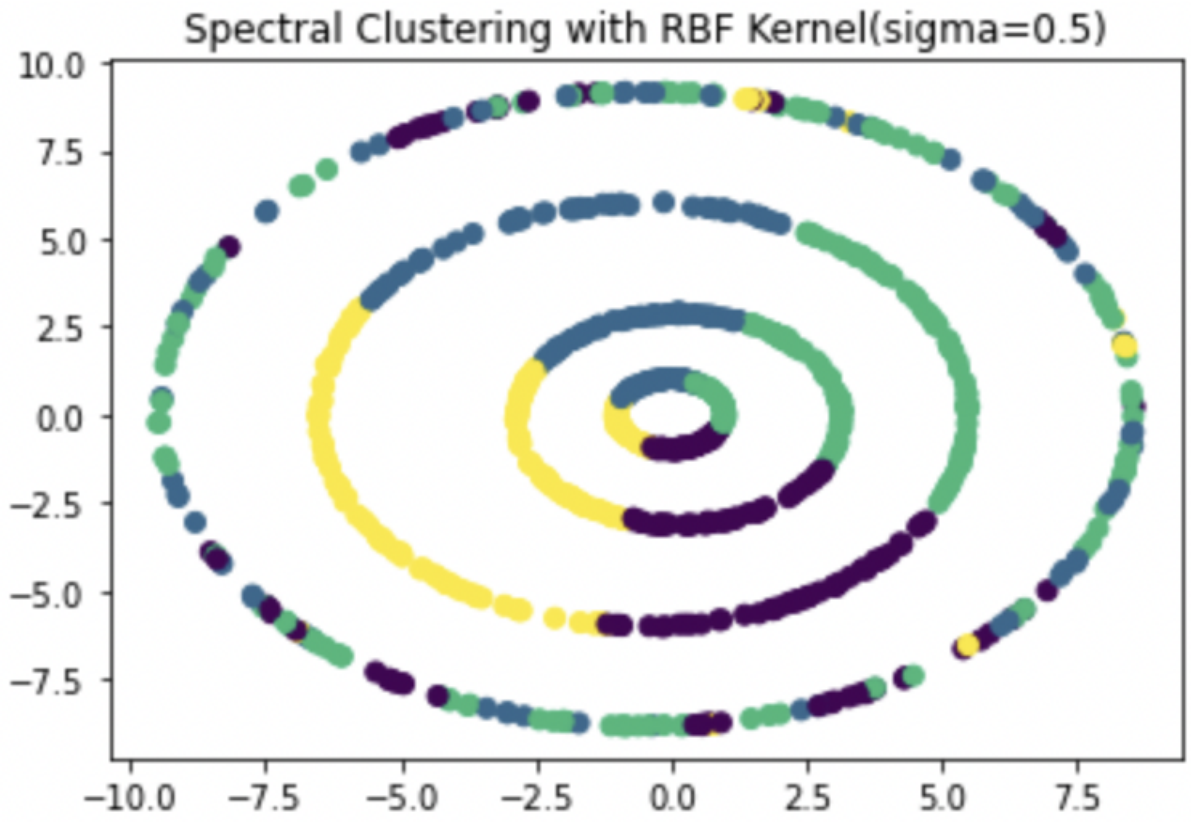
For the RBF Kernel, I tried five values of sigma - 0.2, 0.5, 0.8, 1, 1.5.
The clusters and the error function plots are given below.



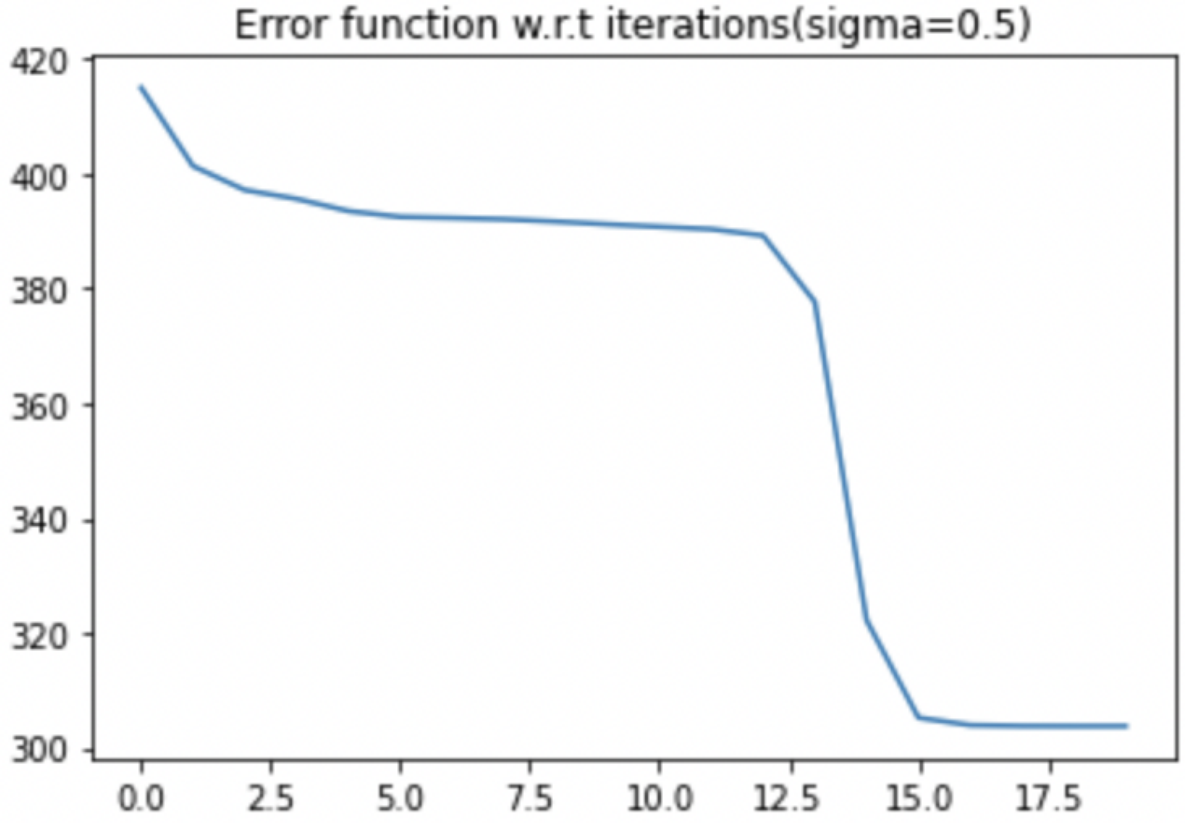
Y axis(vertical) and X axis(horizontal)



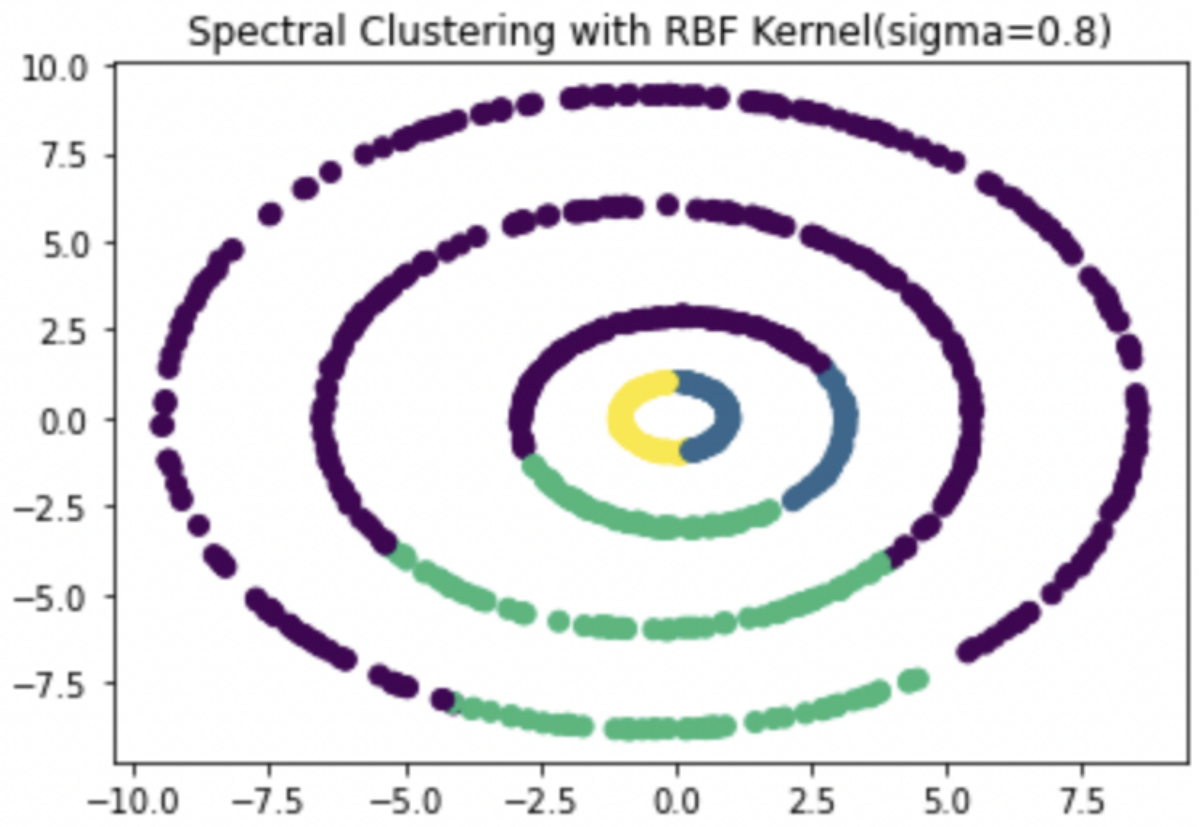
Error (Y axis) vs Iteration Number (X axis)



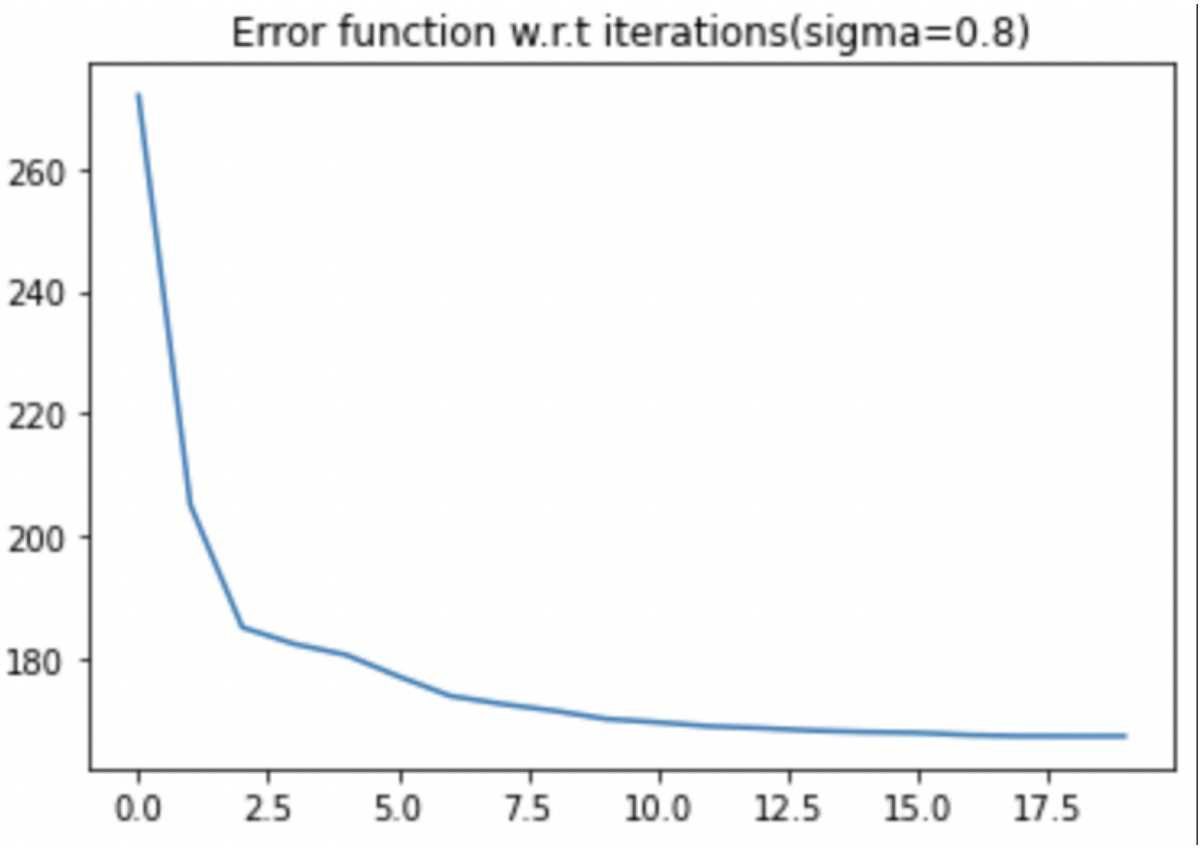
Y axis(vertical) and X axis(horizontal)



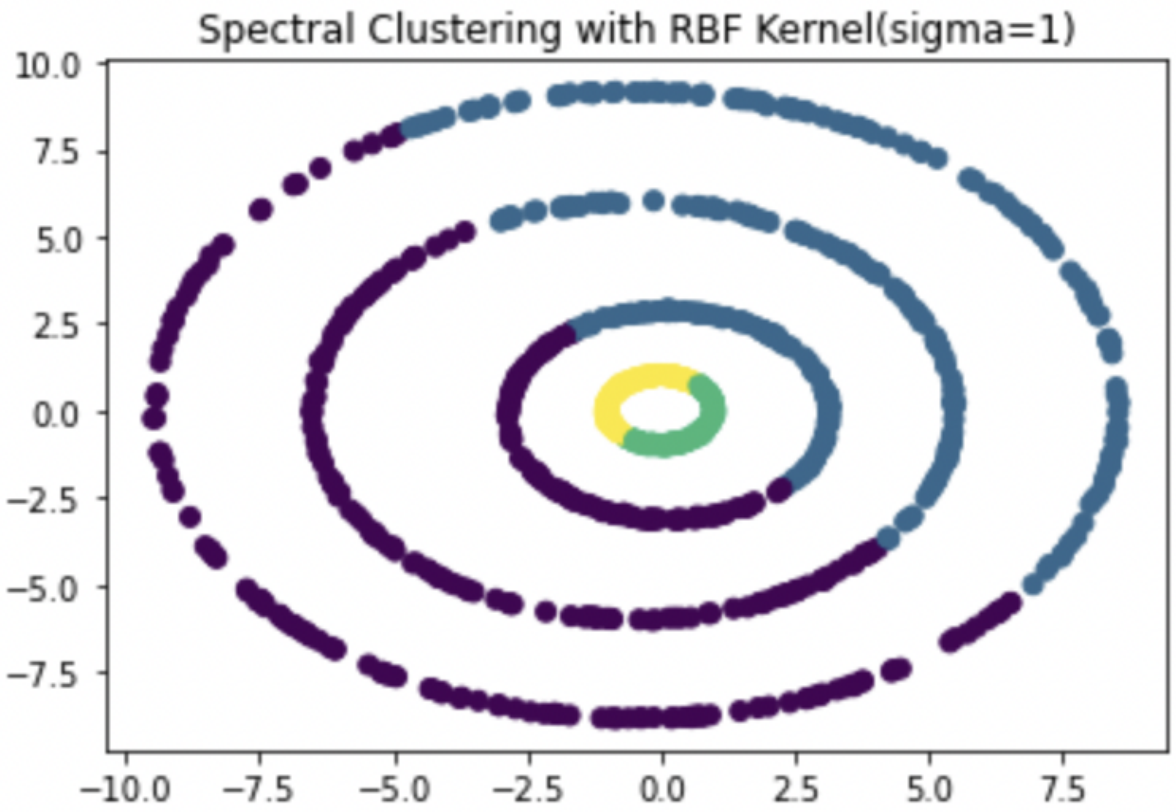
Error (Y axis) vs Iteration Number (X axis)



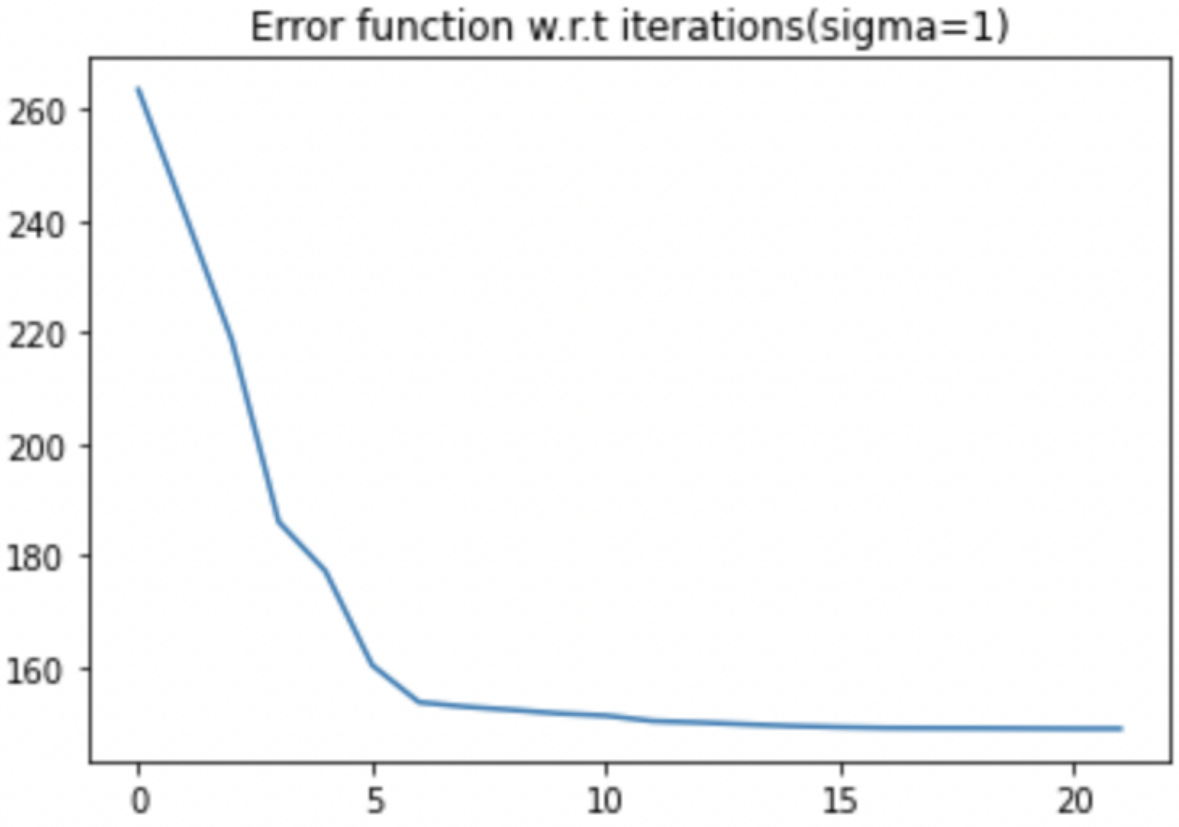
Y axis(vertical) and X axis(horizontal)



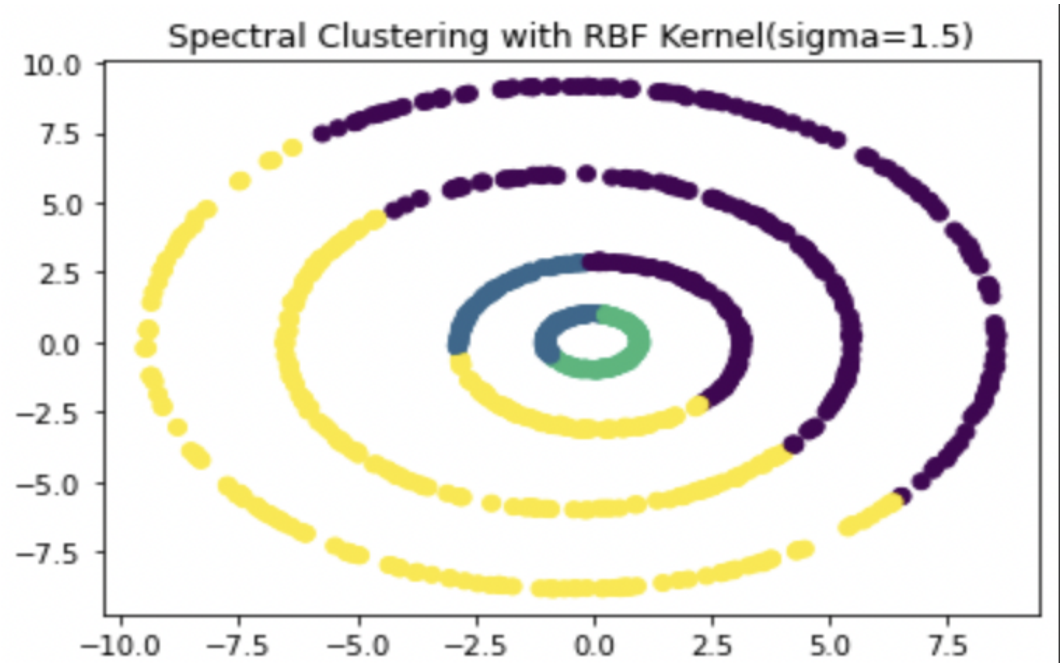
Error (Y axis) vs Iteration Number (X axis)



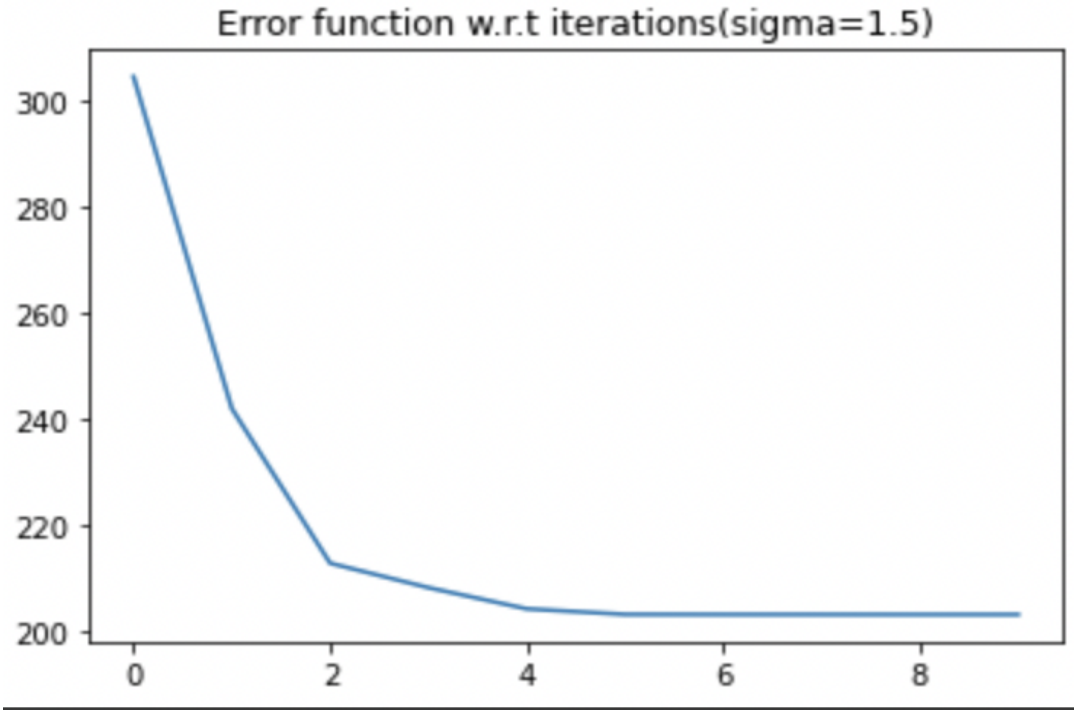
Y axis(vertical) and X axis(horizontal)



Error (Y axis) vs Iteration Number (X axis)



Y axis(vertical) and X axis(horizontal)



Error (Y axis) vs Iteration Number (X axis)

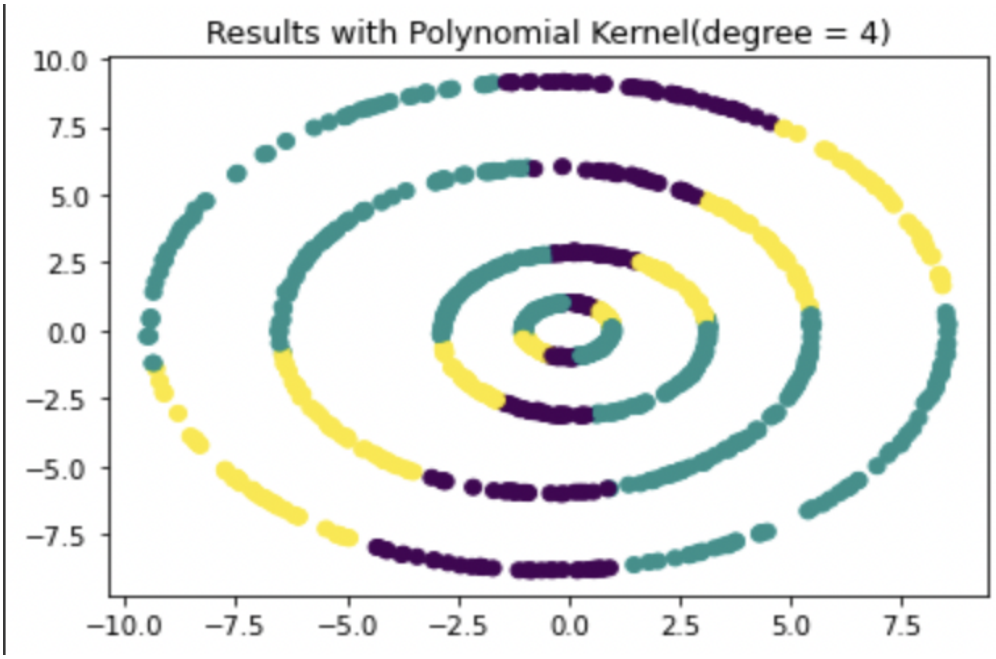
It is clear from the error function plots that the error kept on decreasing as we gradually increased sigma from 0.2 to 1. Then for sigma = 1.5, the error again started increasing. Sigma = 1 gave us the minimum error. So among RBF kernels, sigma = 1 would be my choice for this problem.

In summary, for this problem Polynomial Kernel(degree = 4) and RBF Kernel(sigma = 1) would be my choices.

iv)

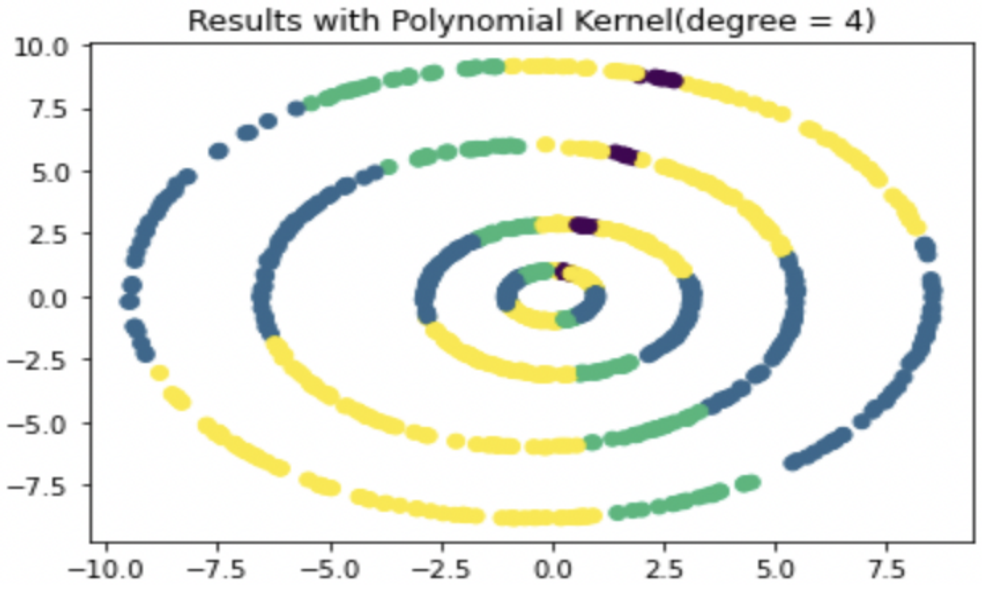
I tried the suggested alternative mapping using both Polynomial Kernel and RBF Kernel.

For the Polynomial Kernel I noticed a general pattern. I experimented with degrees = 2, 3, 4, 5. Most of the time the mapping produced three clusters each with a high number of data points instead of four clusters. One such plot is the following figure.



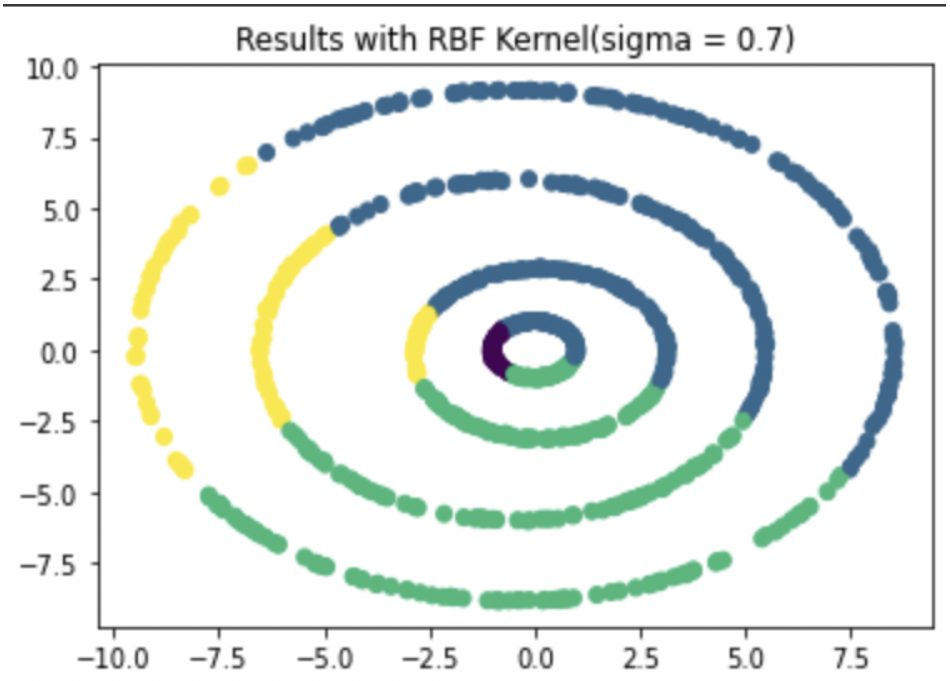
Y axis(vertical) and X axis(horizontal)

Sometimes I did get a fourth cluster with very few data points. The following figure is an example of such a case.



Y axis(vertical) and X axis(horizontal)

For the RBF Kernel also I obtained similar results. One such plot is the following figure.



Y axis(vertical) and X axis(horizontal)

Here also clearly we have three clusters with a high number of data points and the fourth one containing fewer data points.

The clustering problem is NP hard because we are looking for a valid assignment matrix. So, instead what we do is we work with a relaxed version of the problem where we find the top “k” eigenvectors of the kernel matrix. Effectively we consider a (n x k) matrix whose columns are the eigenvectors. Each component of an eigenvector tells us how important the datapoint is for that direction. When we run the K-means algorithm onto this (n x k) matrix considering every row as a data point that gives us spectral clustering. So essentially in spectral clustering we use every component of the eigenvectors in order to obtain the clustering.

The suggested mapping just looks at the maximum eigenvector component corresponding to a data point and assigns that data point to the particular column cluster index. The algorithm does not really use the other eigenvector components which potentially contain directional information about the data points as well.