PATTERN RECOGNITION AND MACHINE LEARNING

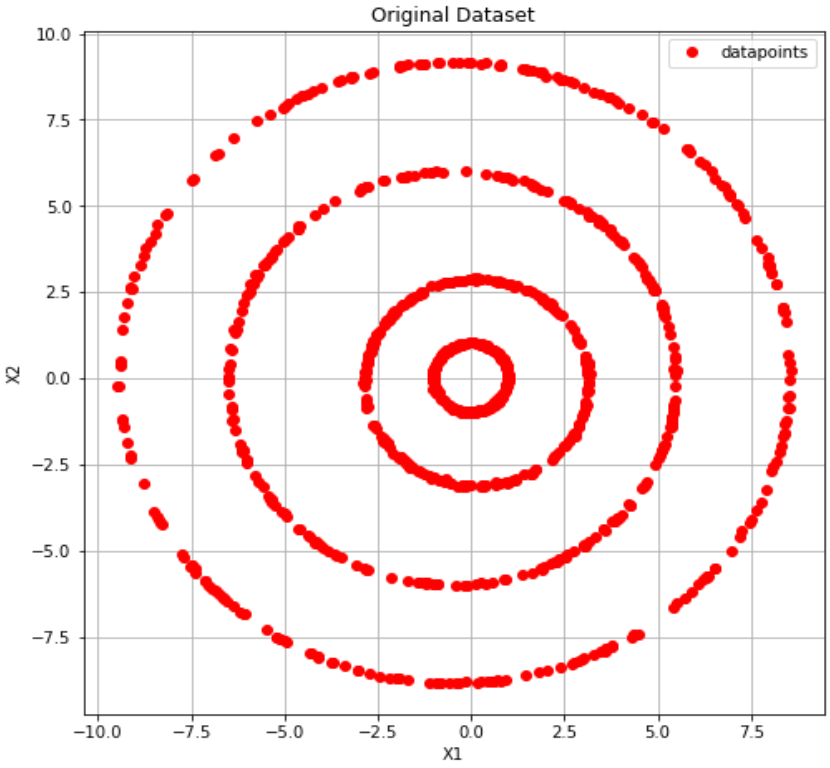
ROLL NO : CS22M084

NAME: SOURADIPTA CHOUDHURI

1.1)

Write a piece of code to run the PCA algorithm on this data-set. How much of the variance in the data-set is explained by each of the principal components?

The code corresponding to this question is in ques-1.py.This is the original dataset. After plotting the data points it looks like 4 concentric circles.This Dataset is a 1000*2 dimensional data. To make is easy I have transposed it and made =2*1000 dimensional data where each column is a datapoint having 2 features.

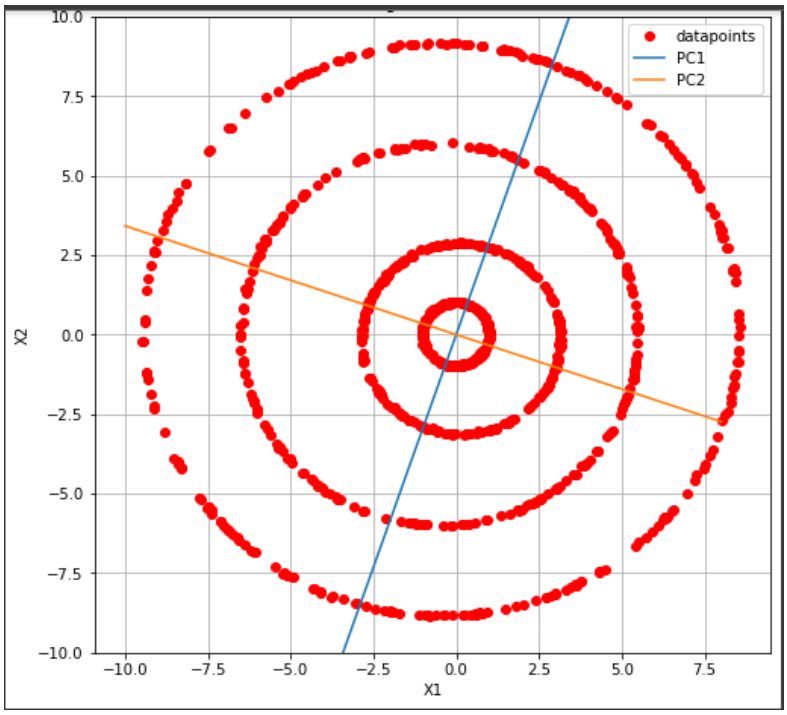


Observation and analysis:

After I run the PCA algorithm on it , I found two eigenvectors and those two eigen vector in [-0.323516 -0.9462227] and [0.9462227 0.323516] and their corresponding eigen values are respectively 17.13(rounded to two decimal places) and 14.49(rounded to two decimal places).

The eigenvalues represent the variance retained by the corresponding eigenvector or the principal component. So here , the eigenvector corresponding to the maximum eigenvalue retains 17.13/(17.13+14.49) *100 %=54.18 % variance and the other eigenvector corresponding to the 2nd eigen value retains 14.49/(17.13+14.49) *100 %= 45.82 %

This diagram is found after plotting both of the eigenvectors =

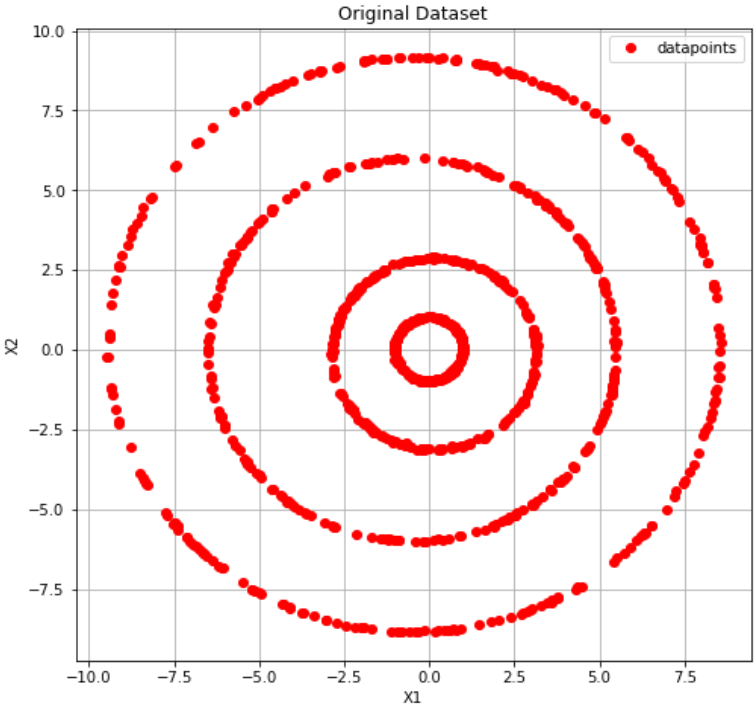


1.2)
Study the effect of running PCA without centering the data-set. What are your observations? Does Centering help?

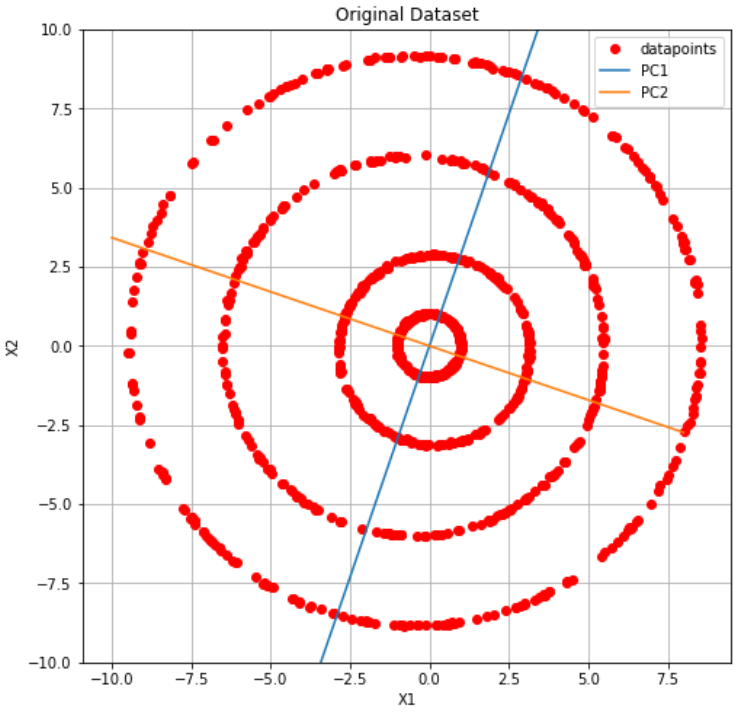
I have tried PCA on both centered and not centered dataset.The code related to uncentered data is in ques-1. Here centering is done by subtracting the mean of the data points. Here every data point has 2 features and the mean of the data points is very low,

it is =[4.075e-07 2.227e-07]. So as the means are very less , even though we do centering , it will not give much effect as these two numbers are negligible.
So after centering also , the dataset remains unchanged and the eigen vector's direction also remains the same. The variance retained by the eigenvectors are also same

The original dataset=



After centering it becomes=



1.3)Write a piece of code to implement the Kernel PCA algorithm on this dataset. Use the following kernels :

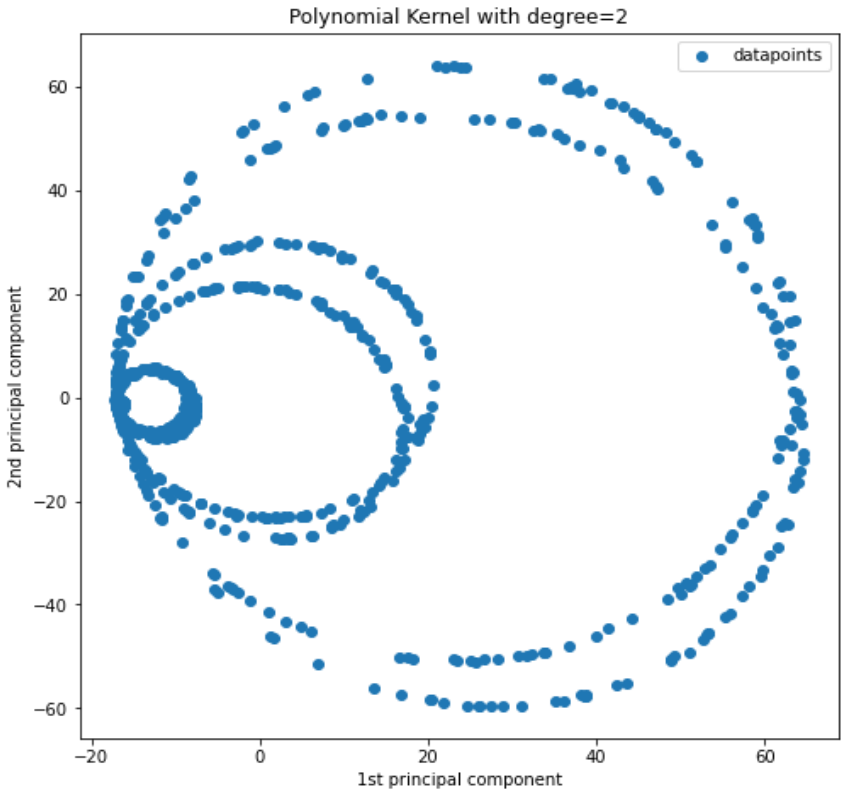
A. $\kappa(x, y) = (1 + x^T y)^d$ for $d = \{2, 3\}$

B. $\kappa(x, y) = \exp \frac{-(x-y)^T(x-y)}{2\sigma^2}$ for $\sigma = \{0.1, 0.2, \ldots, 1\}$

Plot the projection of each point in the dataset onto the top-2 components for each kernel. Use one plot for each kernel and in the case of (B), use a different plot for each value of σ

The code is in ques-1.py.

When I used polynomial kernel with degree 2 then this graph came=

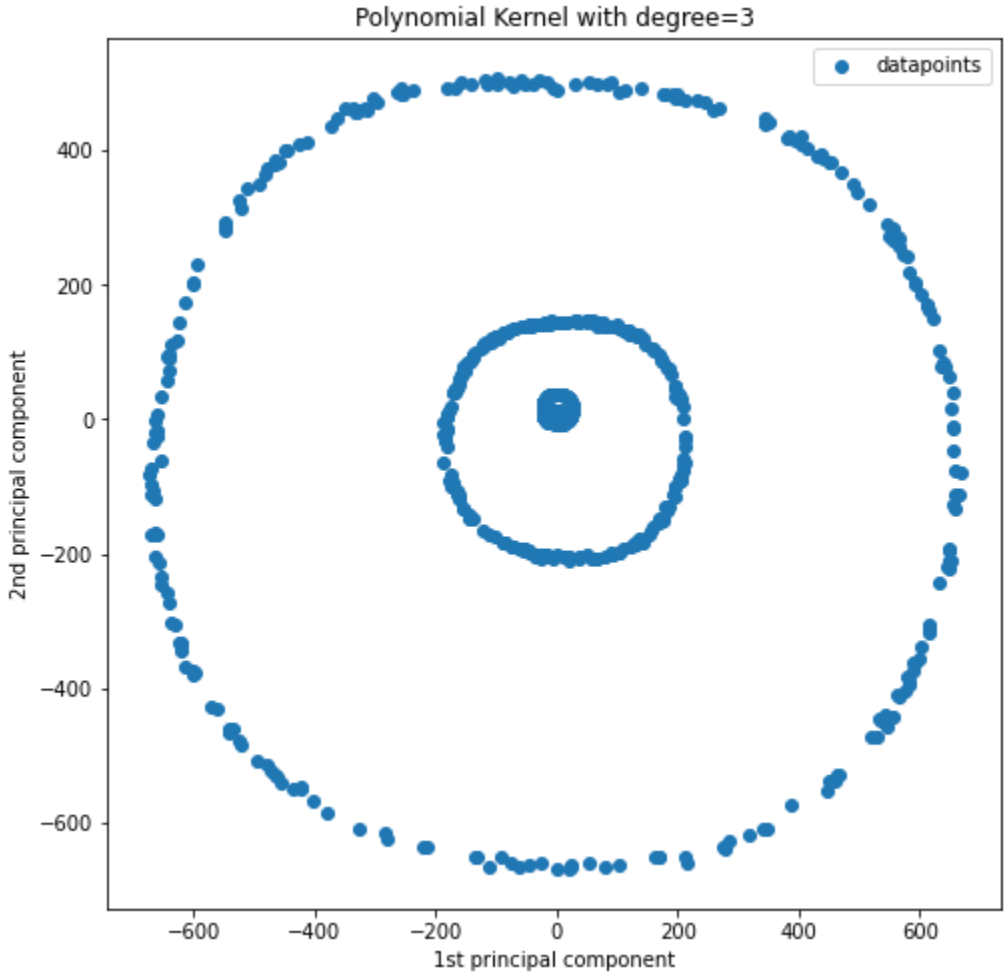


Observation and analysis:

1)There are 5 circles here , and the circles are not centered at the origin now .

2)The circles are shifted from the origin and It is plotted in the graph.

For polynomial of degree =3 we get the following diagram=

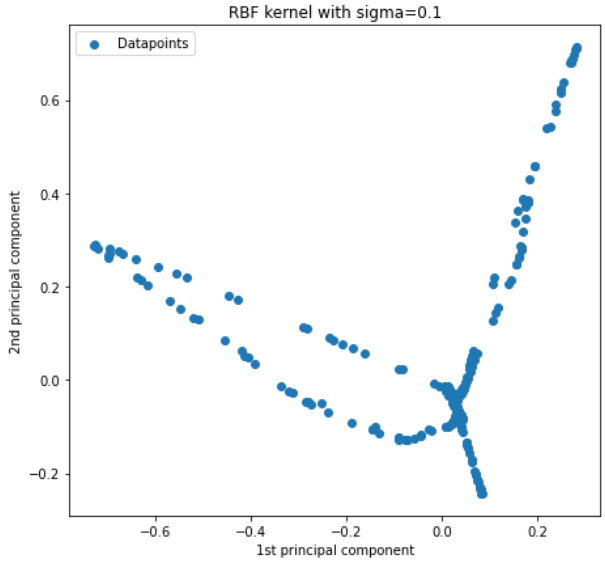


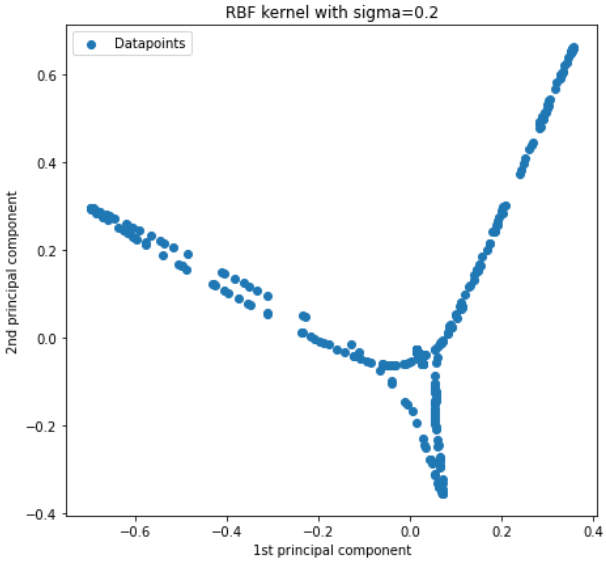
Observation and analysis:

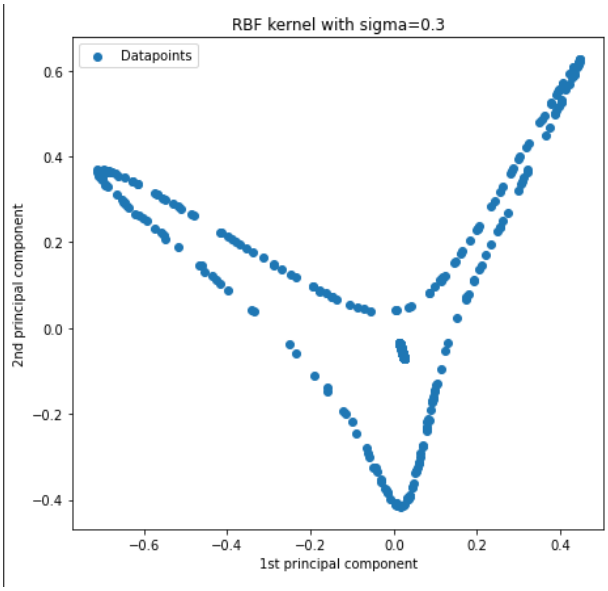
1)Here we can see 3 circles and all of them are centric as their origin is (0,0 point)

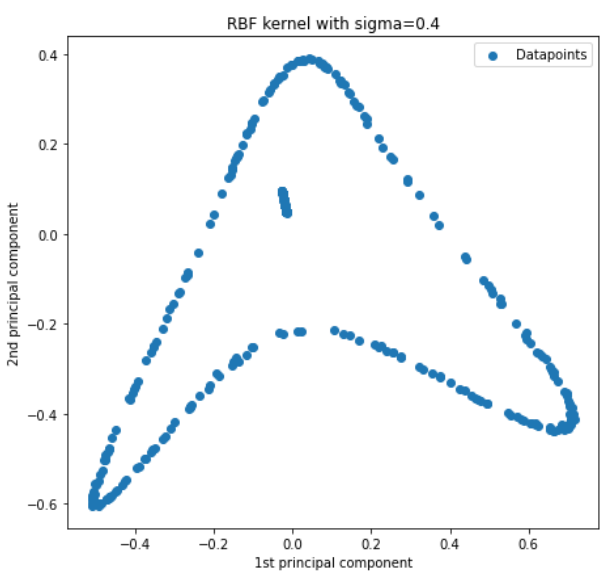
2)This looks similar to the original dataset ,but the original dataset has 4 circles and this graph after reconstruction has 3 circles .

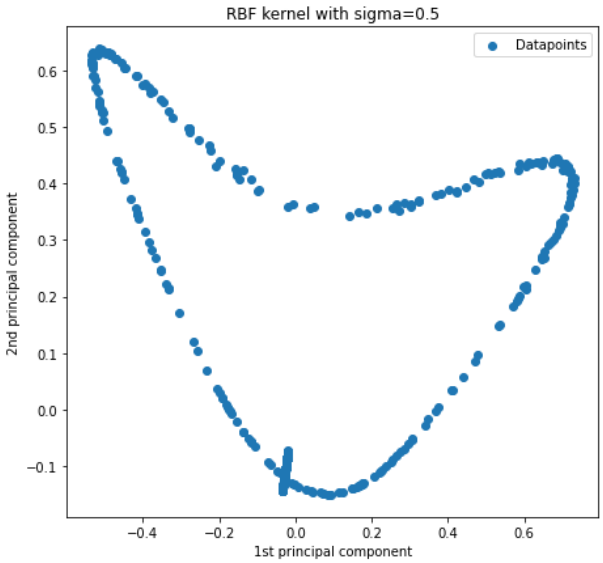
Now I have used the RBF kernel with different sigma values. Like I have tried with sigma values 0.1, 0.2 to 1.0. And in each case the diagram obtained are this=

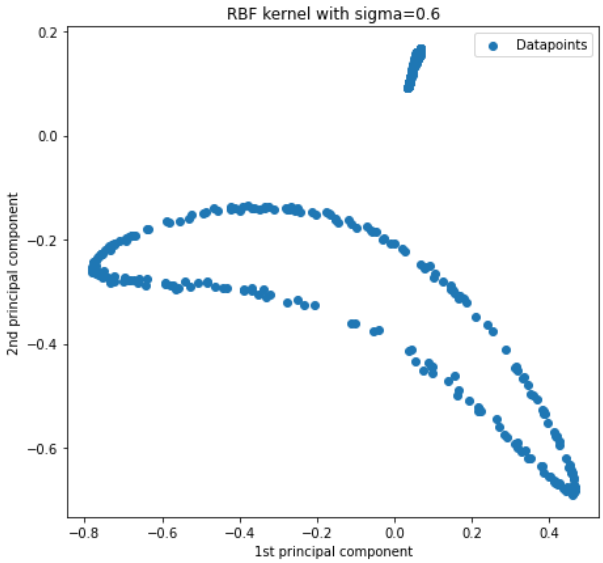

RBF kernel with sigma=0.1
Datapoints
2nd principal component
1st principal component

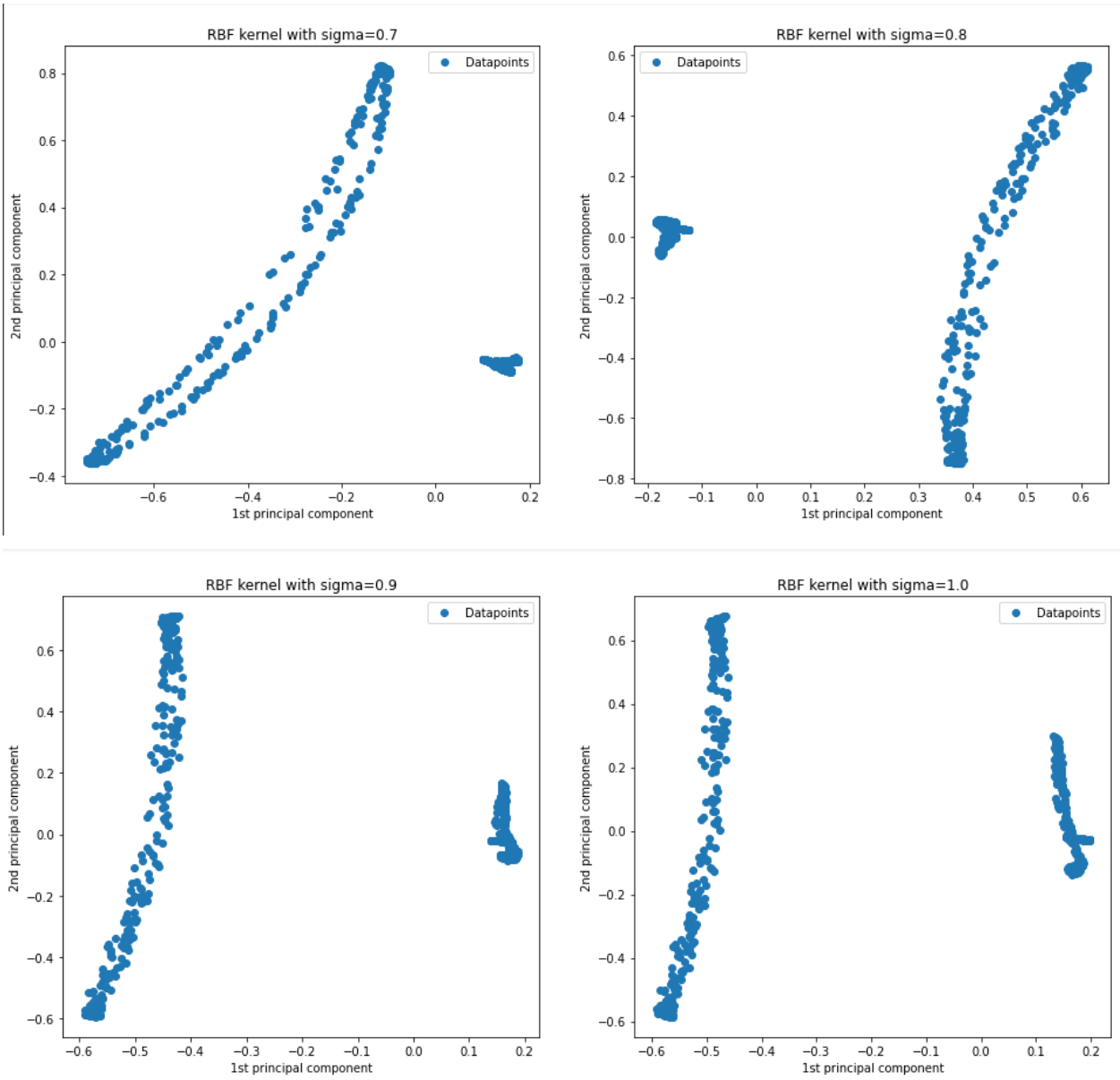

RBF kernel with sigma=0.2
Datapoints
2nd principal component
1st principal component


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


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


RBF kernel with sigma=0.5
Datapoints
2nd principal component
1st principal component
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.4
−0.2
0.0
0.2
0.4
0.6


RBF kernel with sigma=0.6
Datapoints
2nd principal component
1st principal component
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4



Observation and analysis:

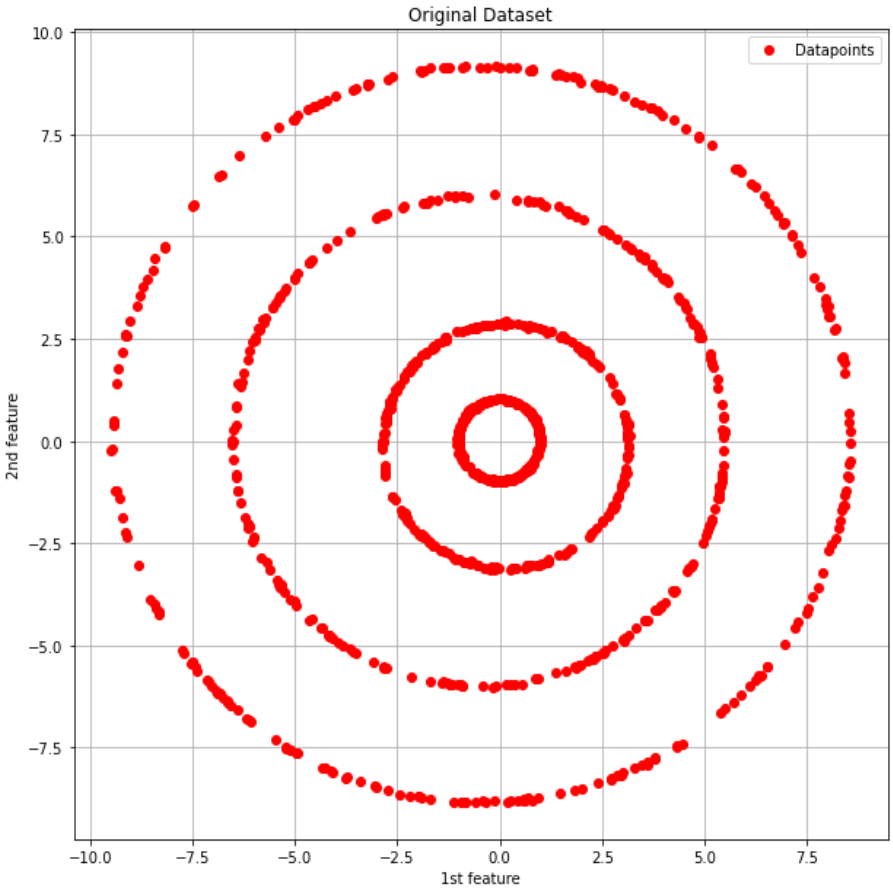
1)No clear structure of data can be seen in any diagram , the circles are coming as different twisted forms .
2)In all diagrams at max two circles can be seen ,while the original dataset has 4 circles in it .
3) So , data loss is happening with this kernel .

1.4)Which Kernel do you think is best suited for this dataset and why?

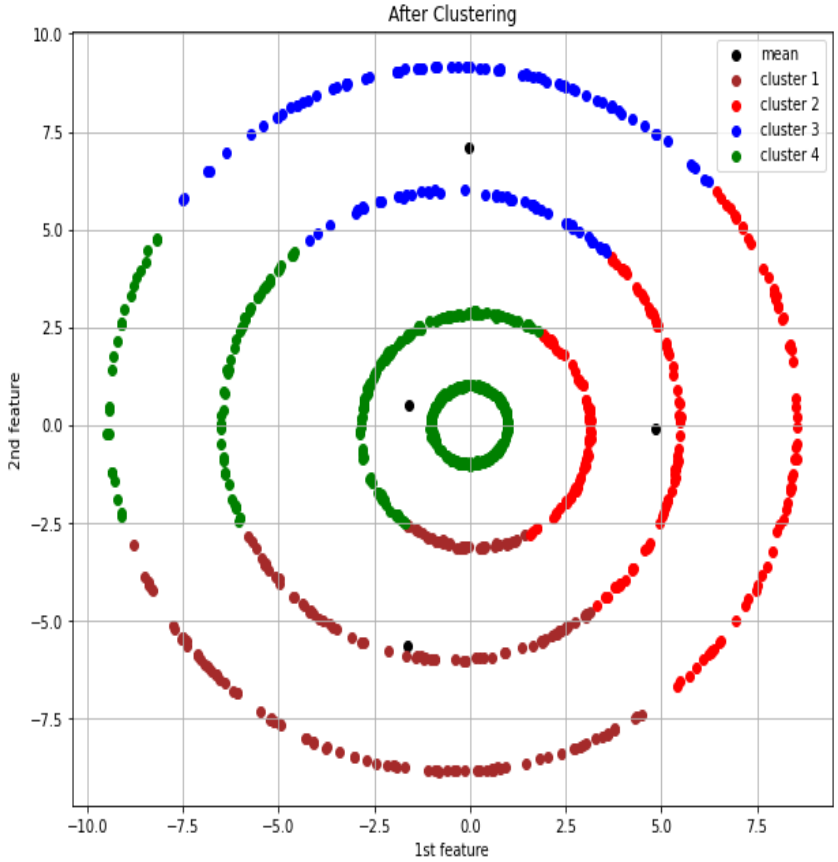
Polynomial kernel with degree 3 works best among all possible solutions. Main purpose of PCA is to reduce the number of components of a large input graph.So, PCA can be used for data compression .
Output of PCA is some principal components . These components are used for data representation .here when we are using polynomial kernel of degree 3 ,then we are creating the reconstruction graph and this graph is more or less similar to the original dataset , but is is not the same when we use other kernels like polynomial kernel with degree 2 and also RBF kernel with different sigma.
So polynomial kernel with degree 3 is the best kernel here

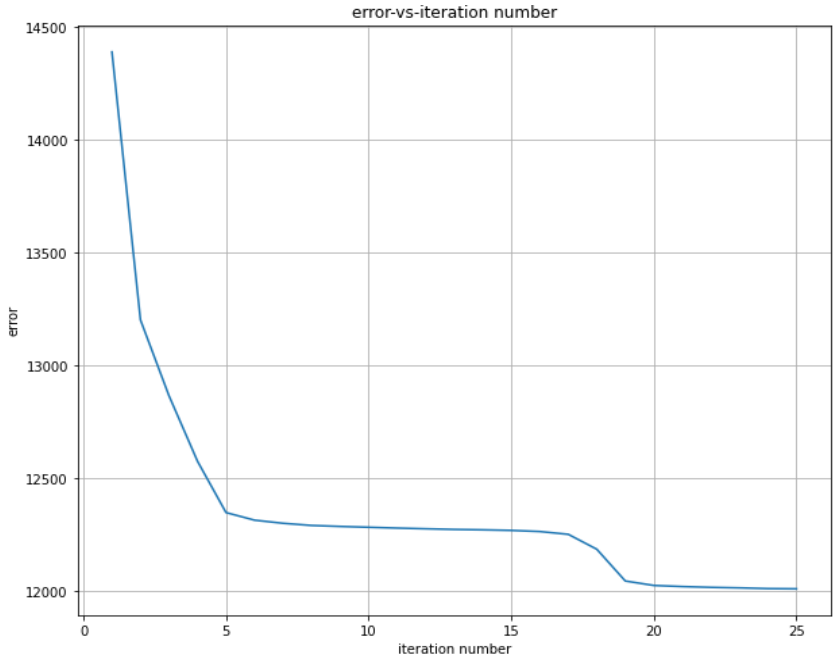
2.Write a piece of code to run the algorithm studied in class for the K-means problem with k = 4 . Try 5 different random initialization and plot the error function w.r.t iterations in each case. In each case, plot the clusters obtained in different colors.

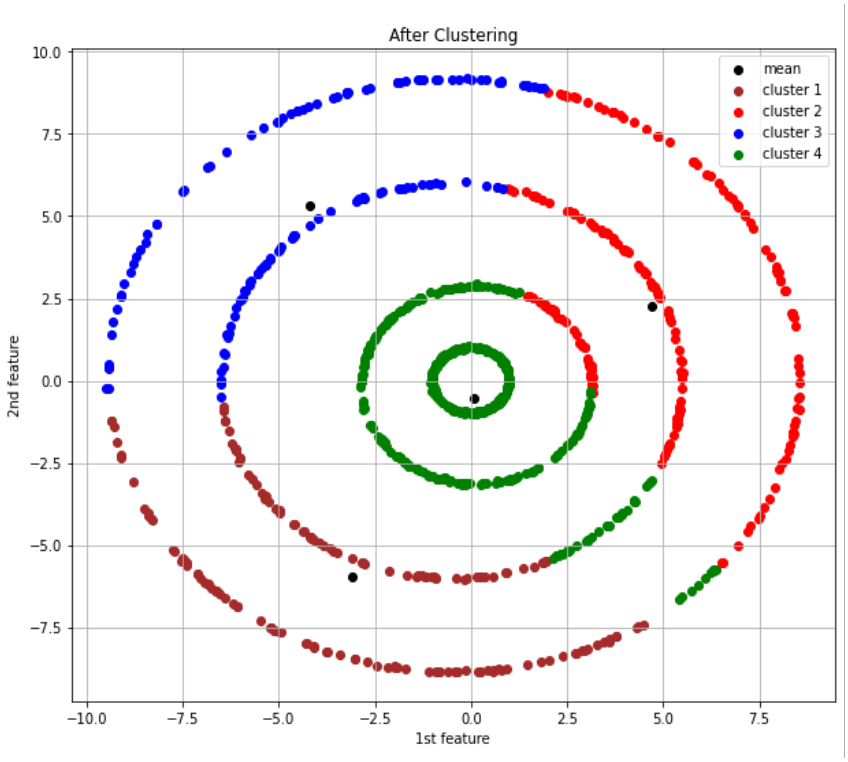
In this question ,the same dataset of the previous question is used.The dataset is 4 concentric circles of different radius.

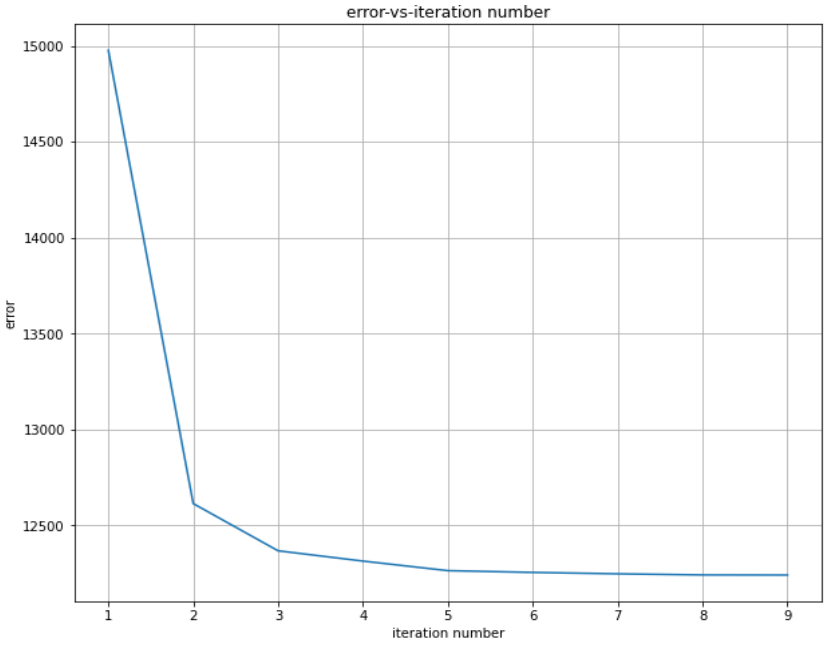


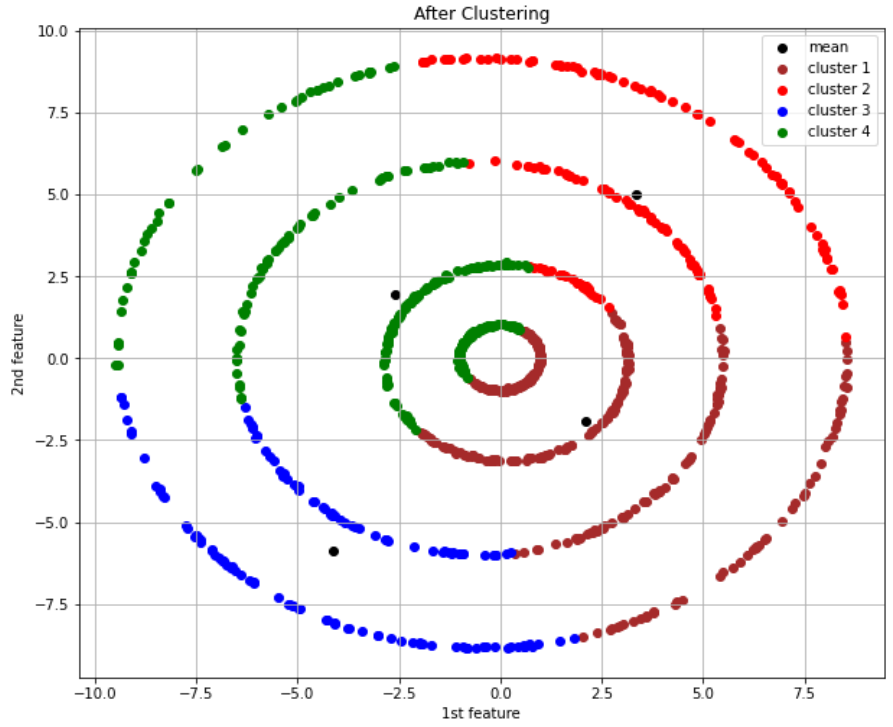
The code for this question is in ques-2.py.I have used 5 different initializations . I have plotted the data points after clustering and also the error function with the iteration number for each of those initializations=

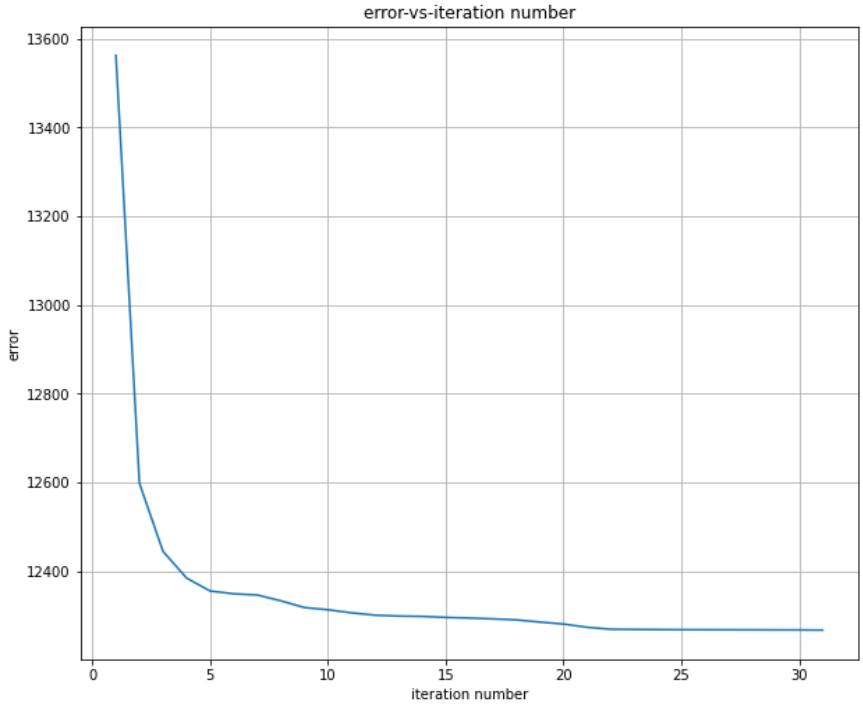

After Clustering
mean
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

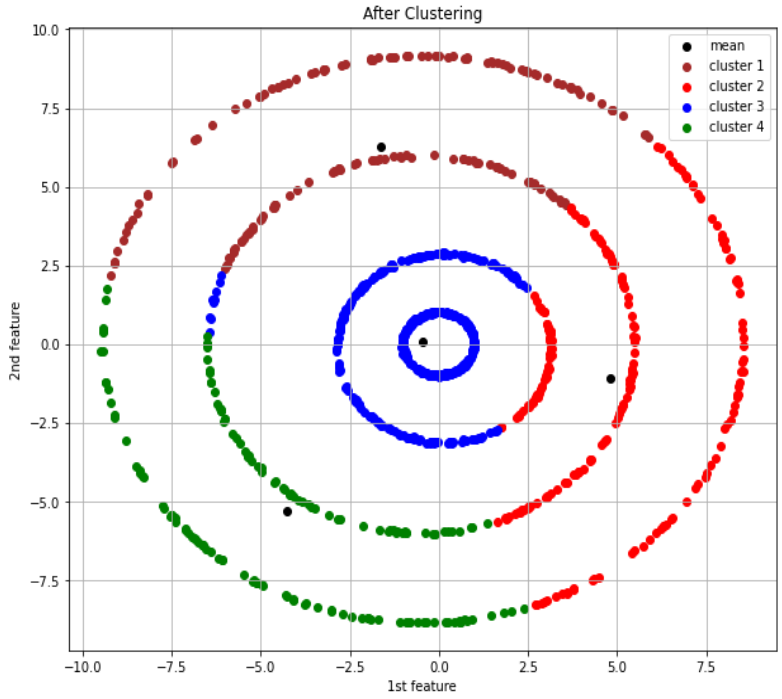

error-vs-iteration number
error
iteration number
14500
14000
13500
13000
12500
12000
0
5
10
15
20
25

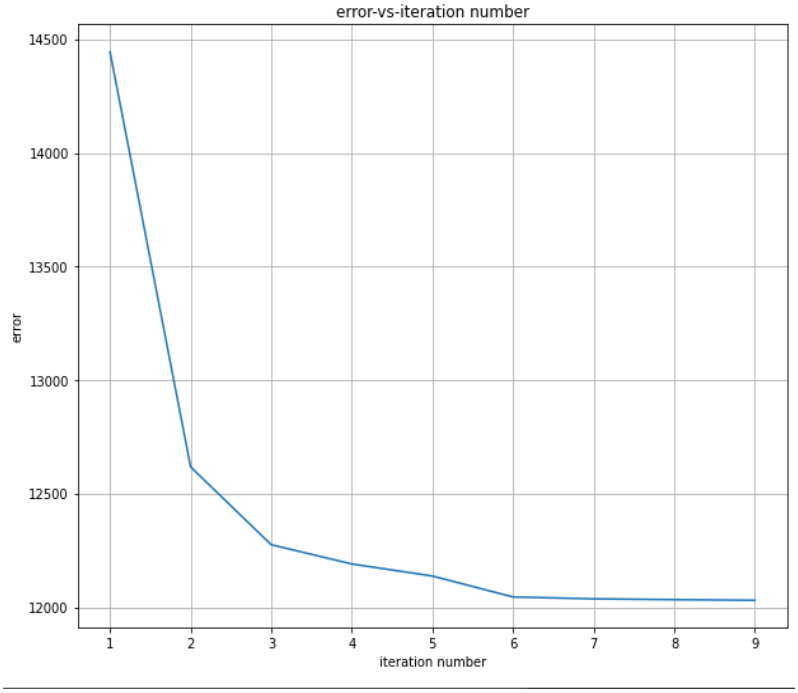

After Clustering
mean
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

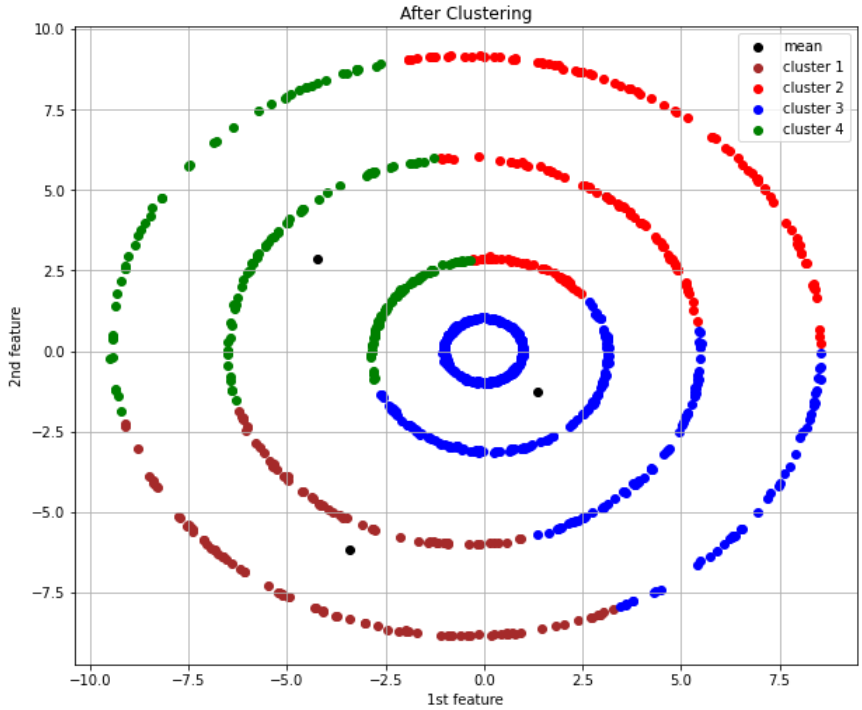

error-vs-iteration number
error
iteration number
15000
14500
14000
13500
13000
12500
1
2
3
4
5
6
7
8
9

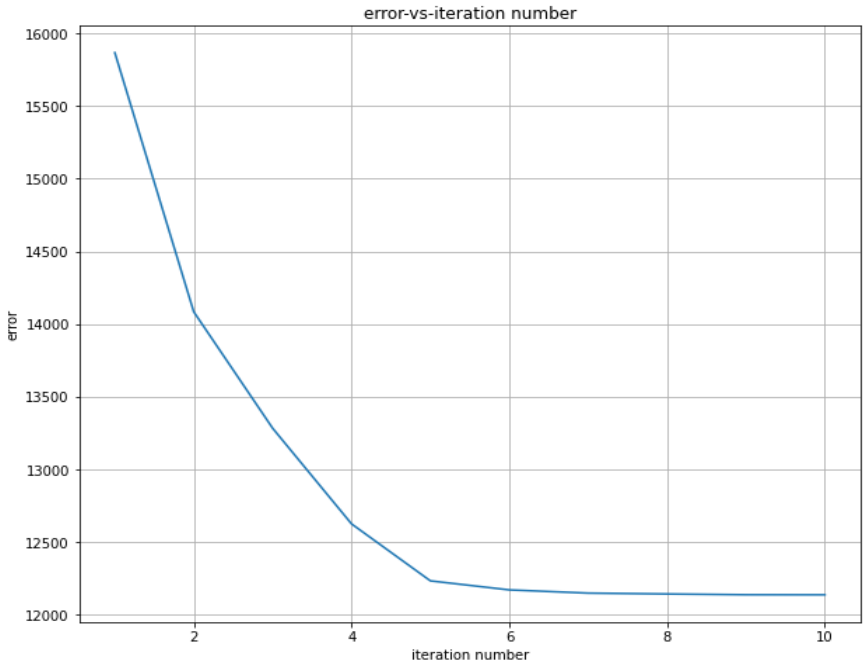

After Clustering
mean
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5


error-vs-iteration number
error
iteration number
13600
13400
13200
13000
12800
12600
12400
0
5
10
15
20
25
30


After Clustering
mean
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5


error-vs-iteration number
error
iteration number
14500
14000
13500
13000
12500
12000
1
2
3
4
5
6
7
8
9


After Clustering
mean
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

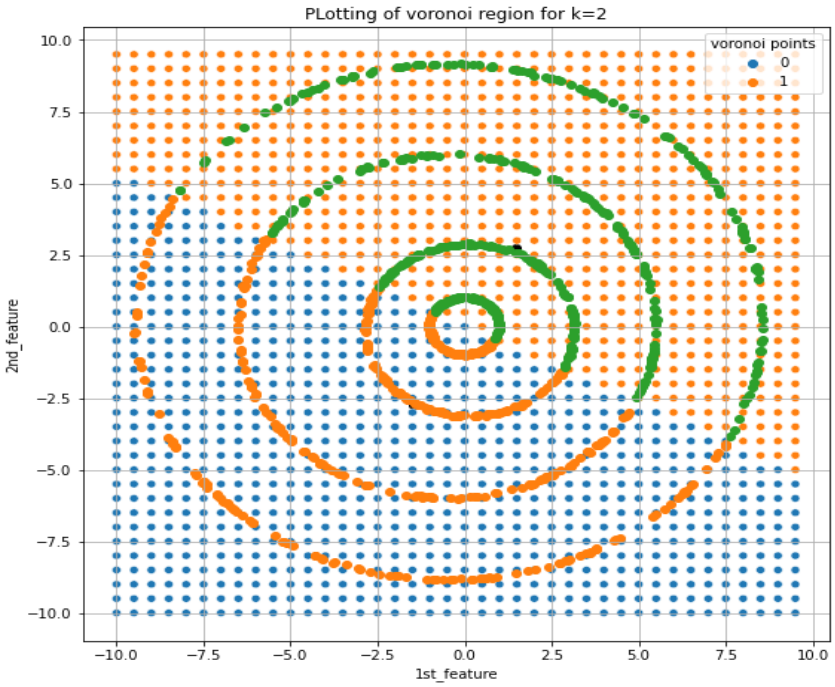

error-vs-iteration number
error
iteration number
16000
15500
15000
14500
14000
13500
13000
12500
12000
2
4
6
8
10

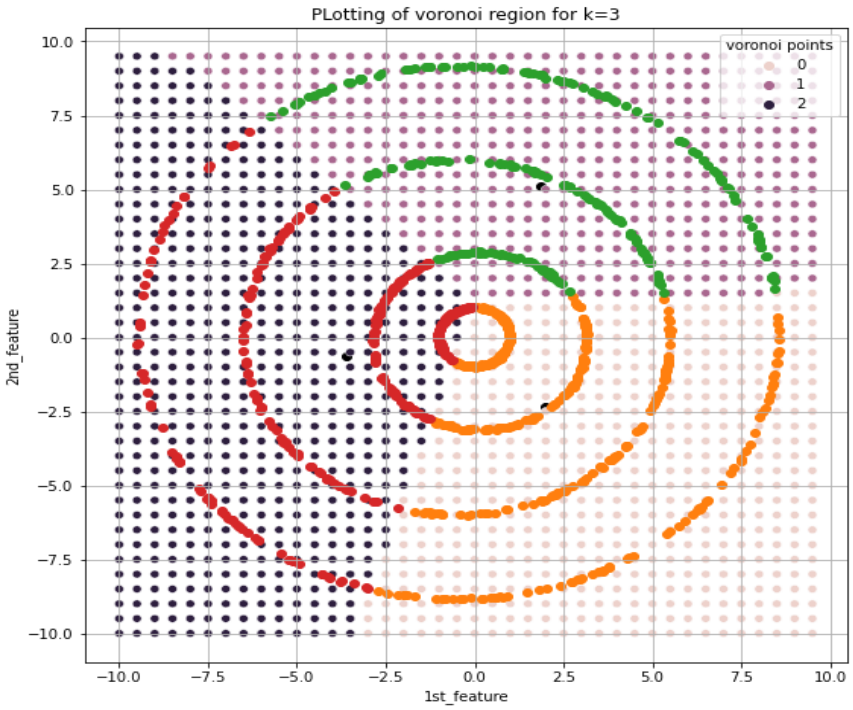
Observation and analysis:

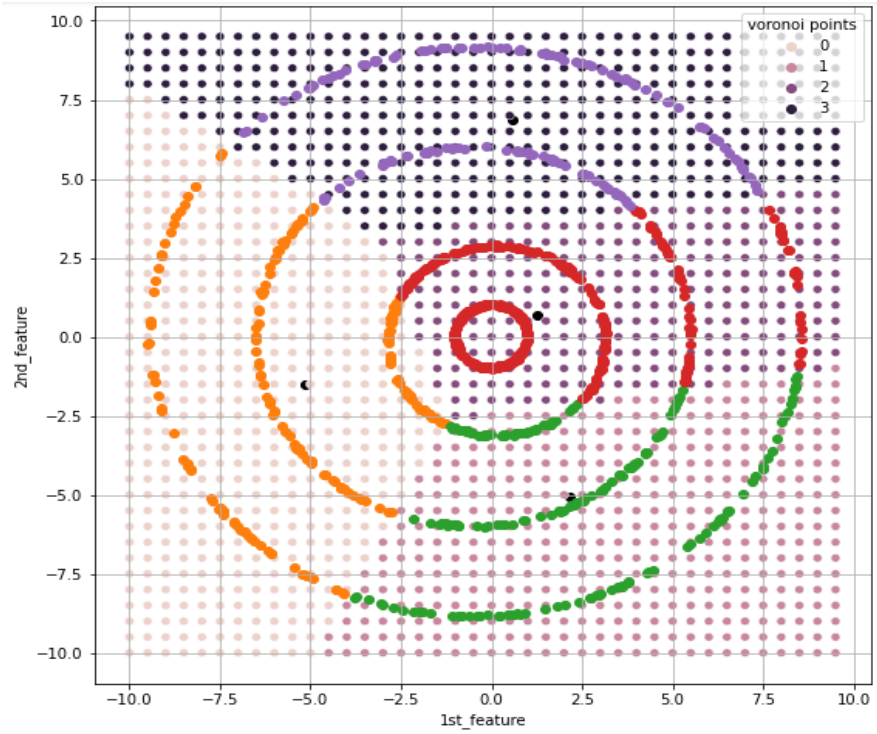
At first the error value decreases for every iteration . But sometimes it does not decrease any longer . We call it the convergence.Here after convergence the error function value is not 0 in any case . The algorithm has converged; it means the assignments of the points don't change and the error function can not be lower than what is achieved . error function value will be 0 when there are n=k points where k is the number of clusters.

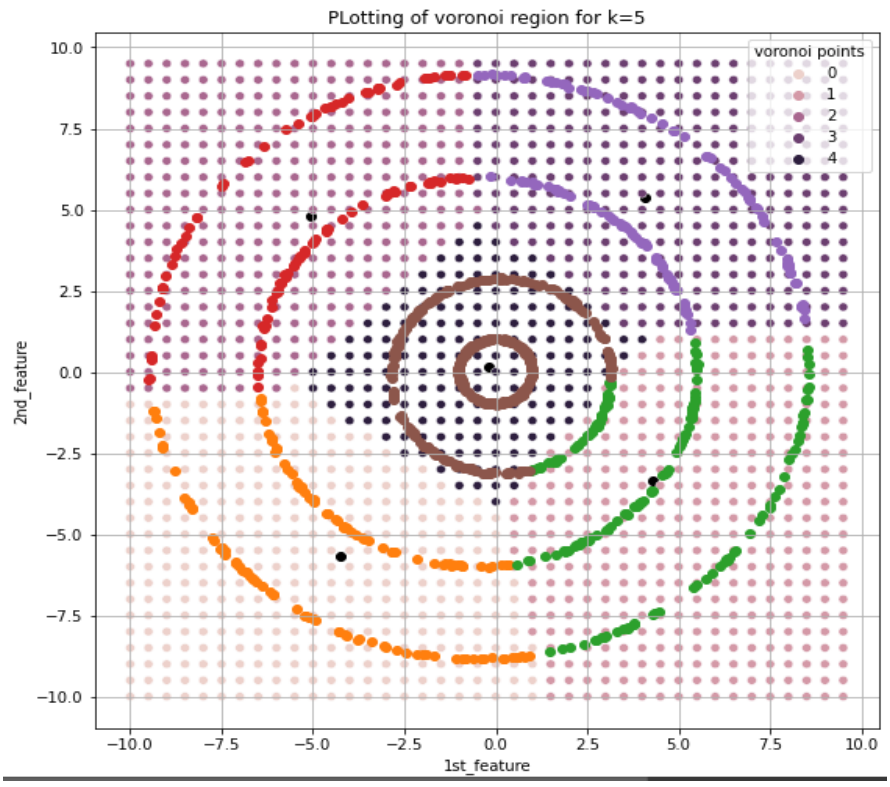
2.2)Fix a random initialization. For K = {2, 3, 4, 5}, obtain cluster centers according to the K-means algorithm using the fixed initialization. For each value of K, plot the Voronoi regions associated with each cluster center. (You can assume the minimum and maximum value in the data-set to be the range for each component of R 2).

For k=2 (when number of cluster is 2)=


PLotting of voronoi region for k=2
voronoi points
0
1
2nd_feature
1st_feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0


PLotting of voronoi region for k=3
voronoi points
0
1
2
2nd_feature
1st_feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0


voronoi points
0
1
2
3
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
2nd_feature
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
1st_feature

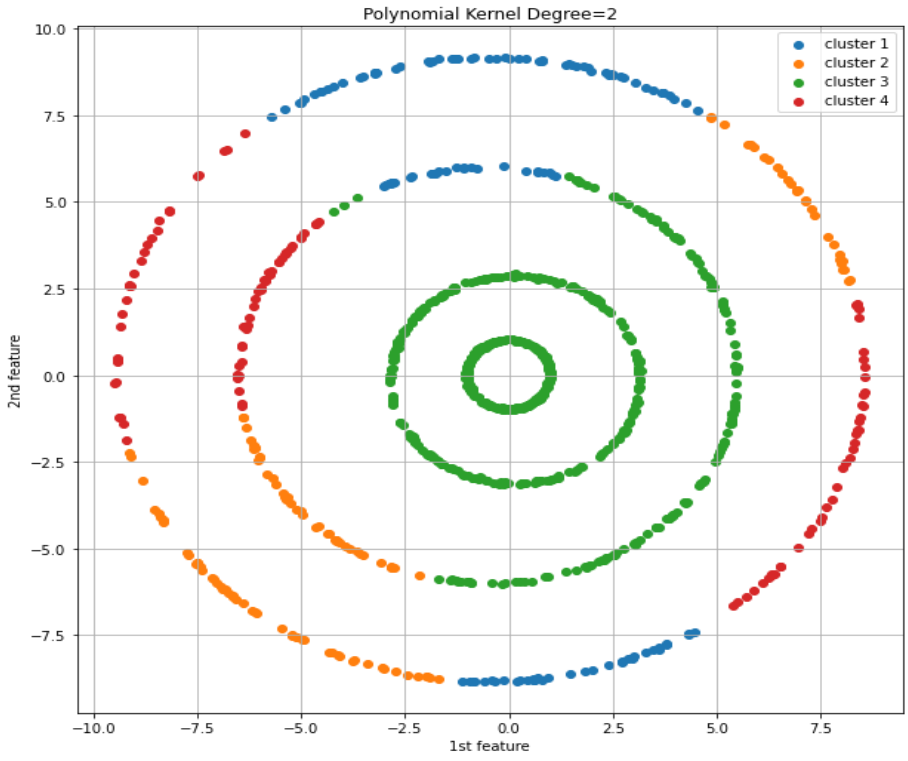

PLotting of voronoi region for k=5
voronoi points
0
1
2
3
4
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
2nd_feature
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
1st_feature

Observation and analysis:

Here k denotes cluster number . If k==2 then it means cluster number is 2. So the number of voronoi regions is 2.voronoi region is a convex region. All circles are divided into multiple regions but natural clusters do not come (here natural cluster means every). For getting natural clusters we use spectral clustering i.e. we go to higher dimensions.

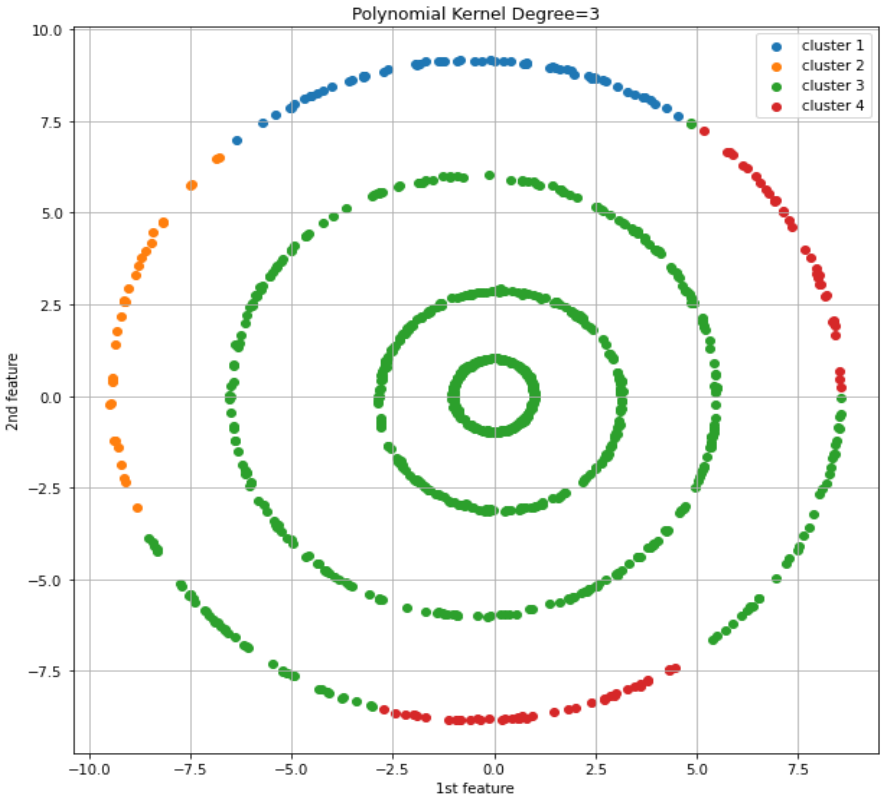
2.3)Run the spectral clustering algorithm (spectral relaxation of K-means using KernelPCA) k = 4. Choose an appropriate kernel for this data-set and plot the clusters obtained in different colors. Explain your choice of kernel based on the output you obtain.

Code for this question is given in ques-2.py file.I have used a polynomial kernel for degree =2 and degree =3, RBF kernel for sigma values between 0.1 to 1.0 giving 0.1 distance between each number.
This is the result after using a kernel matrix made by using polynomial kernel of degree =2

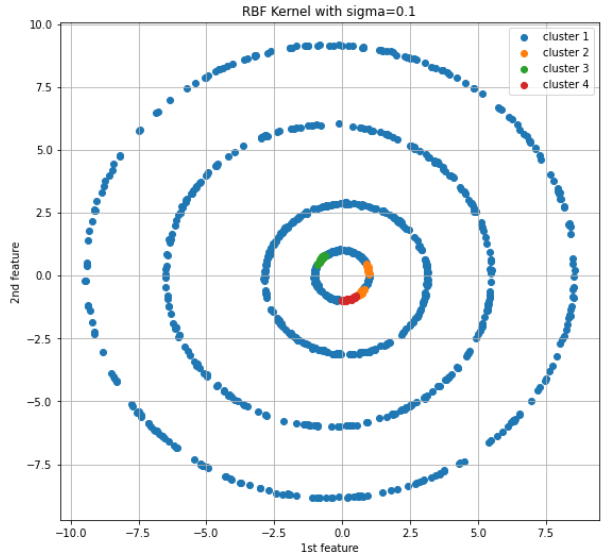


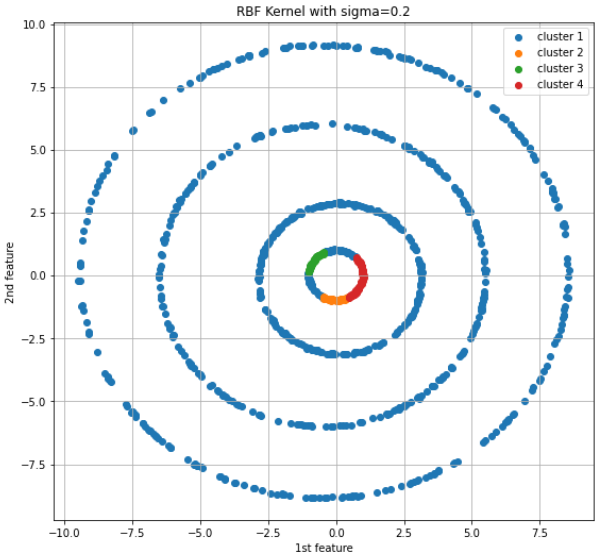
Observation and analysis:
Here also we don't get the natural clusters , but here we may not get any voronoi region like in ques-2.py. 4 clusters are obtained with 4 different colors.

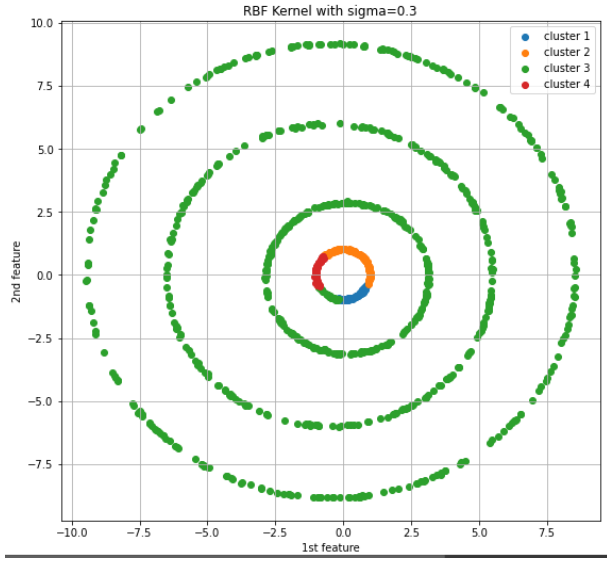
This is the result after using a kernel matrix made by using polynomial kernel of degree =3

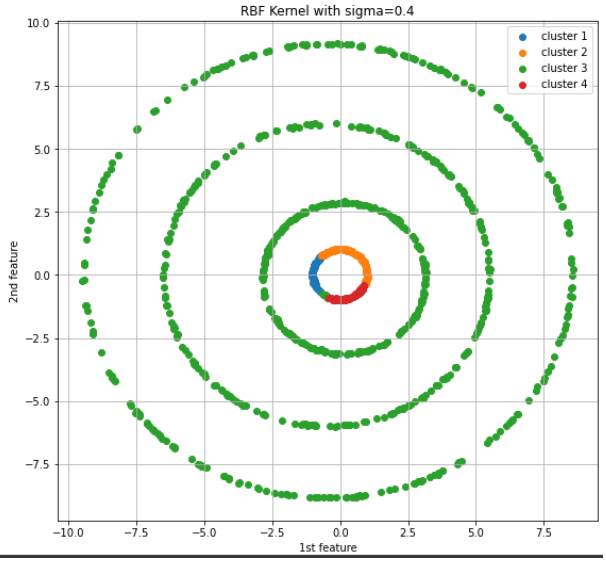


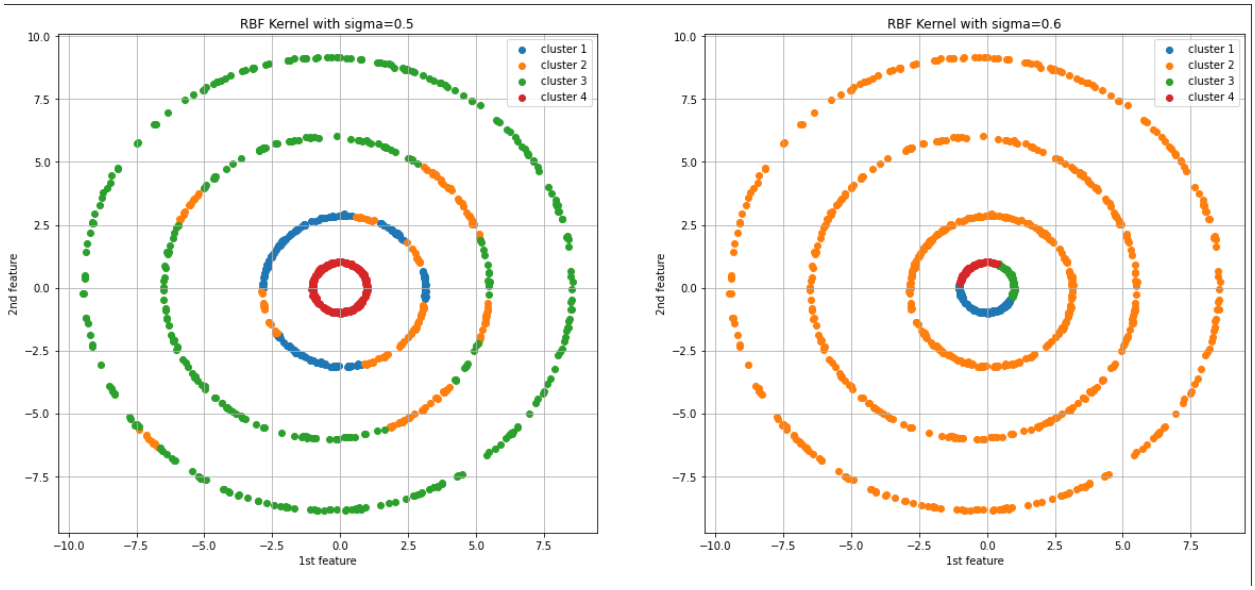
Observation and analysis:
Here also we don't get the natural clusters . 4 clusters are obtained with 4 different colors. but here we may not get any voronoi region like in ques-2.py.
This is the clustering result after using a kernel matrix made by using RBF kernel with sigma value in the range=[0.1 ,0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

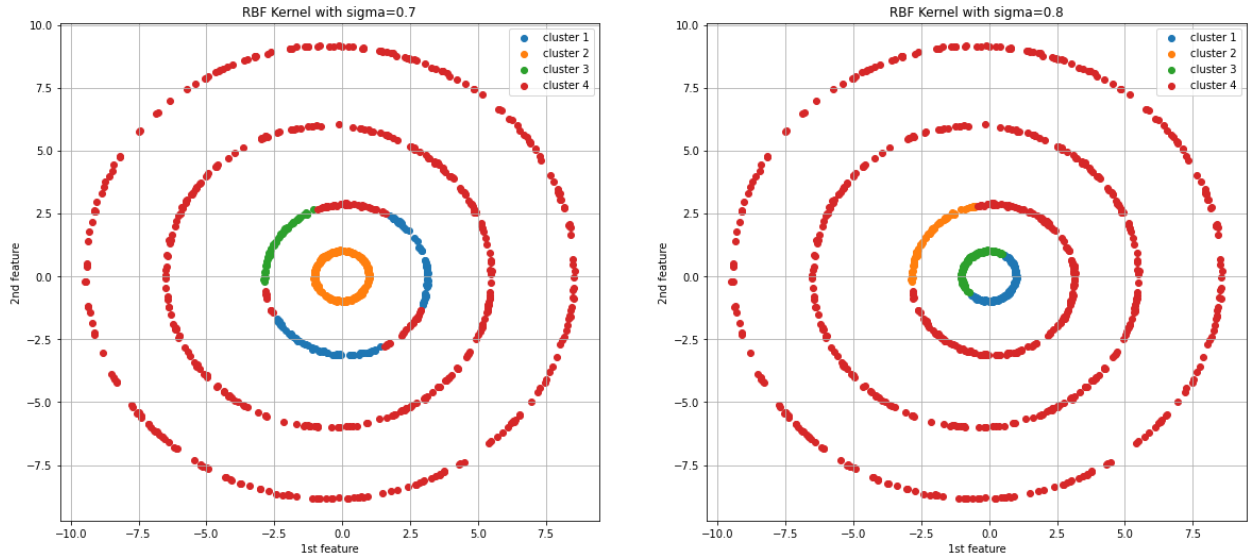

RBF Kernel with sigma=0.1
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature

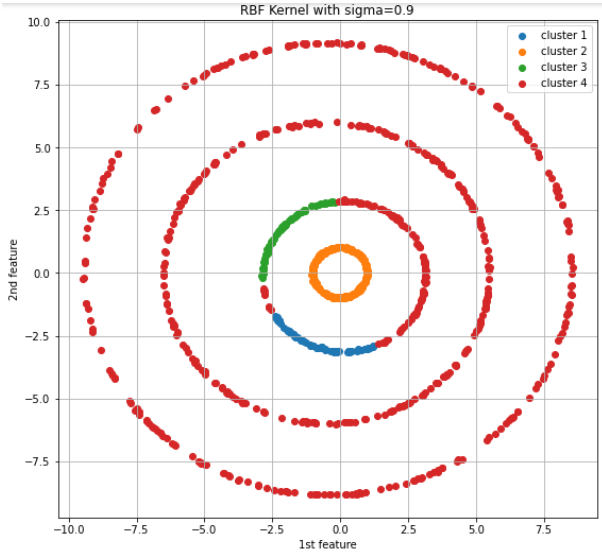

RBF Kernel with sigma=0.2
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature

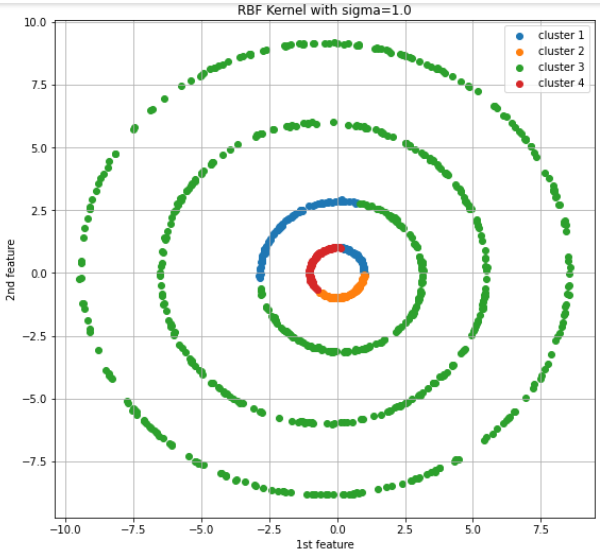

RBF Kernel with sigma=0.3
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature


RBF Kernel with sigma=0.4
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature


RBF Kernel with sigma=0.5
RBF Kernel with sigma=0.6
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature


RBF Kernel with sigma=0.7
RBF Kernel with sigma=0.8
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature





Observation and analysis:

The RBF kernel does not work as good as the polynomial kernel with degree 2 and polynomial kernel with degree 3. Here also we don't get the natural clusters .Even 2 to 3 circles(data points in the circles) may belong to the same cluster . We may not get any voronoi region .

Choosing the best kernel:

<u>In all these below lines, error values are the final objective function value i.e. objective function value when k means algorithm converges.</u>
As if I run the k-means one time and find the error value , there may be some dependency because of initialization , so I have run k-means multiple times and then found the average of all error values.

- After these steps , I have found a sample average of 10 error values with a kernel function of **polynomial degree =2**. It came out to be=1.889(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a kernel function of **polynomial degree =3**. It came out to be=2.617(rounded to 3 decimal places)

- I have found a sample average of 10 error values with a **sigma value of 0.1** in RBF kernel function. It came out to be=1.889(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.2** in RBF kernel function. It came out to be=1.630(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.3** in RBF kernel function. It came out to be=1.547(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.4** in RBF kernel function. It came out to be=1.660(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.5** in RBF kernel function. It came out to be=1.885(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.6** in RBF kernel function. It came out to be=2.418(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.7** in RBF kernel function. It came out to be=2.126(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.8** in RBF kernel function. It came out to be=1.982(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 0.9** in RBF kernel function. It came out to be=1.670(rounded to 3 decimal places)
- I have found a sample average of 10 error values with a **sigma value of 1.0** in RBF kernel function. It came out to be=1.496(rounded to 3 decimal places)

So from all of these values , it can be seen that RBF with a sigma value of 1.0 is a better option (other than that , polynomial 2 also gives good results). But in general , RBF is a better choice than polynomial kernel because for many sigma values , RBF performs better than polynomial kernel with degree 2.

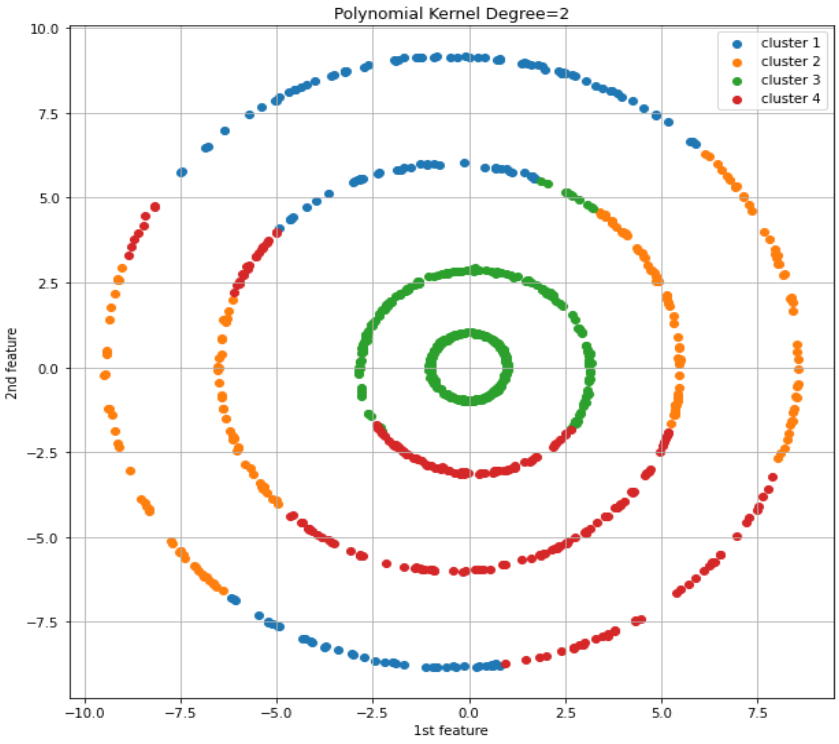
2.4)Instead of using the method suggested by spectral clustering to map eigenvectors to cluster assignments, use the following method: Assign data point i to cluster ` whenever

$$\ell = \arg \max_{j=1,\ldots,k} v_i^j$$

where $v^j \in \mathbb{R}^n$ is the eigenvector of the Kernel matrix associated with the j-th largest eigenvalue. How does this mapping perform for this dataset?. Explain your insights

Code for this question is given in ques-2_part-4.py file.I have used a polynomial kernel for degree =2 and degree =3, RBF kernel for sigma values between 0.1 to 1.0 giving 0.1 distance between each number.

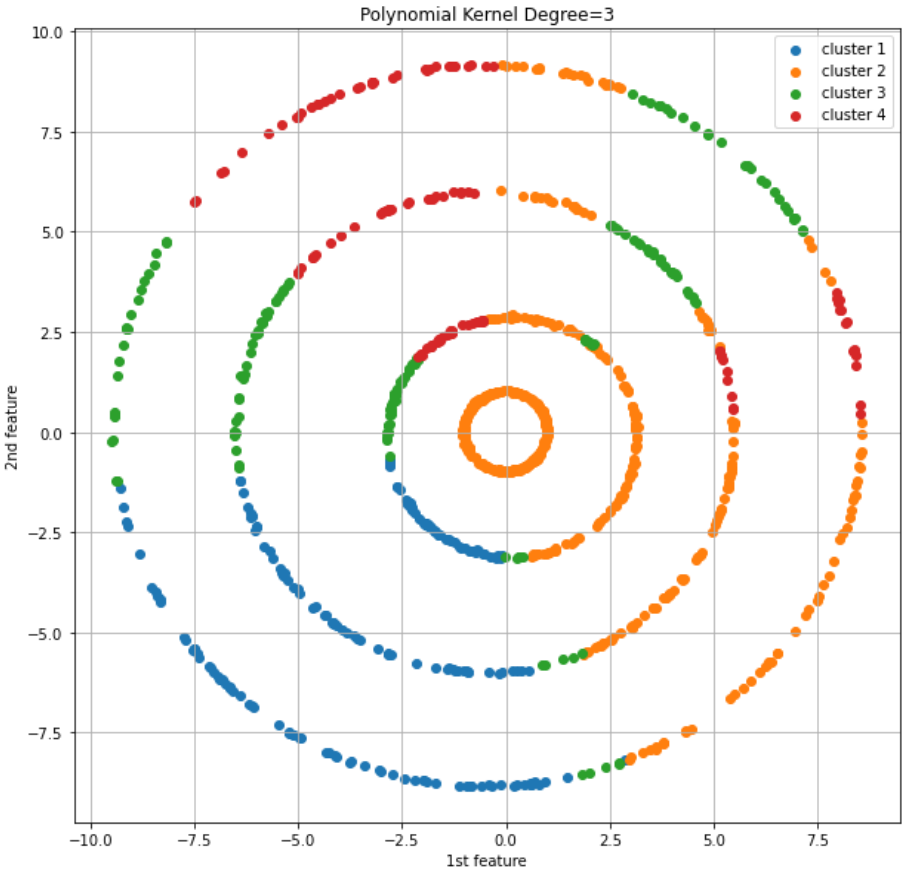
This is the result after using a kernel matrix made by using polynomial kernel of degree =2



Observation and analysis:

Here also we don't get the natural clusters , We may not get any voronoi region like in ques-2.py. 4 clusters are obtained with 4 different colors.

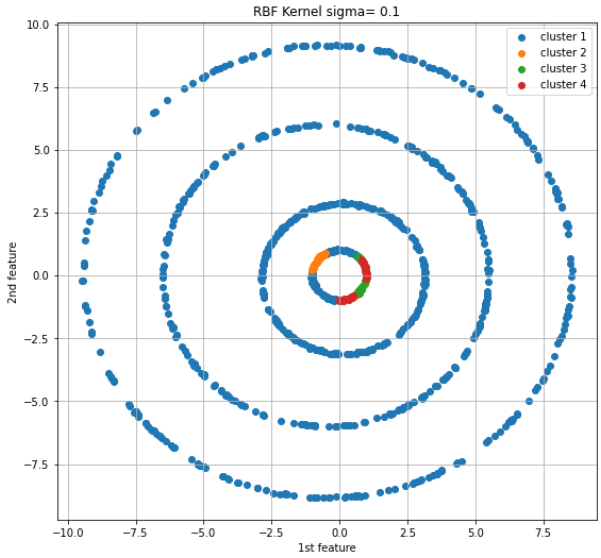
This is the result after using a kernel matrix made by using polynomial kernel of degree =3

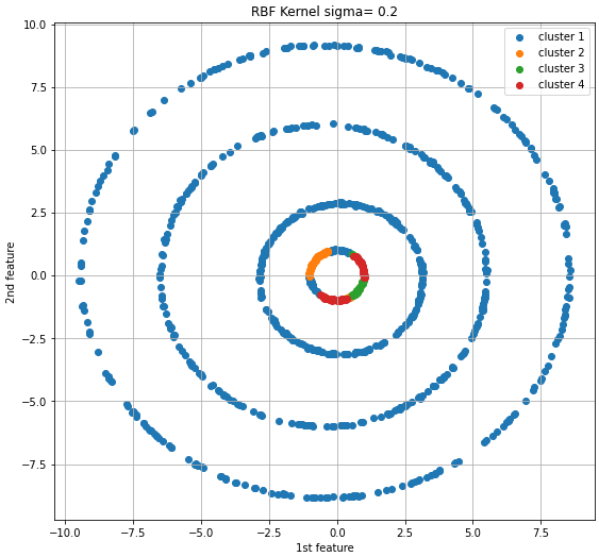


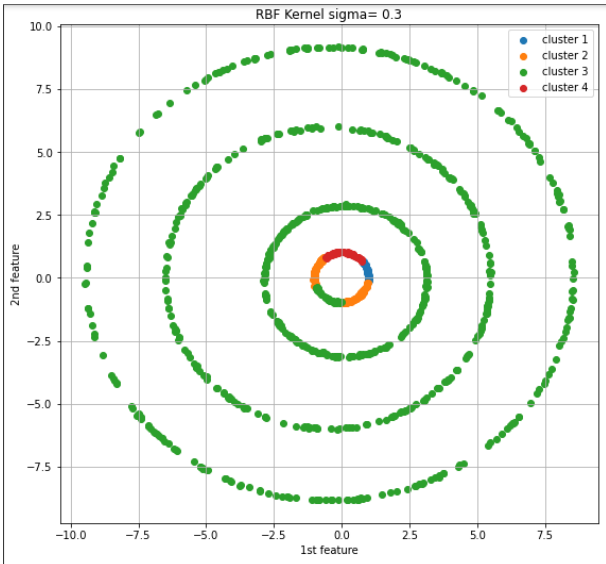
Observation and analysis:

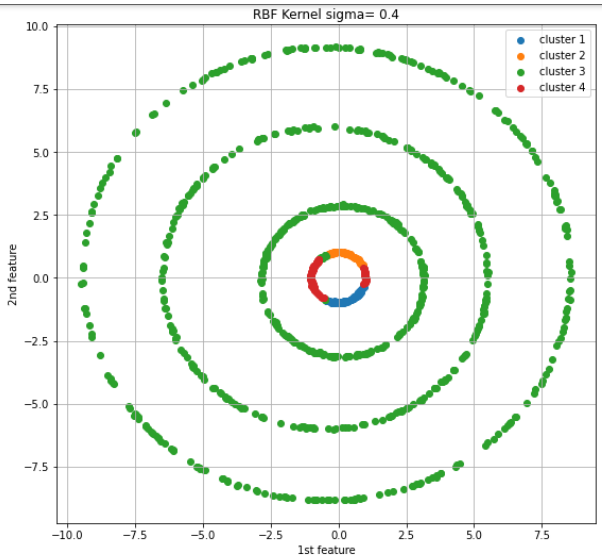
Here also we don't get the natural clusters and we may not get any voronoi region like in ques-2.py. 4 clusters are obtained with 4 different colors.

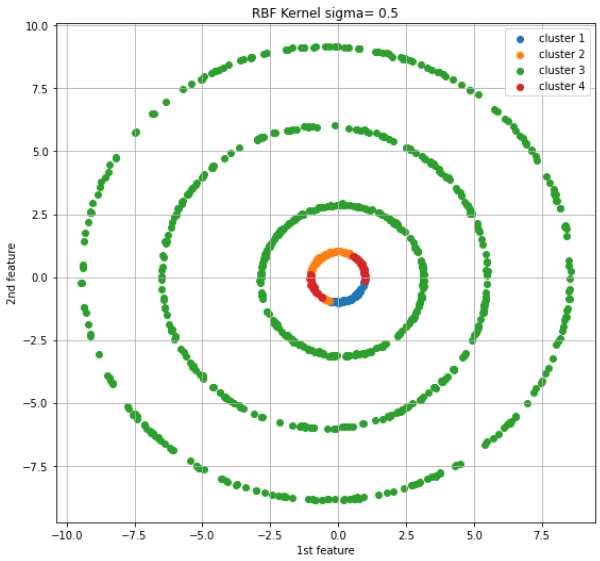
This is the clustering result after using a kernel matrix made by using RBF kernel with sigma value in the range=[0.1 ,0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]=

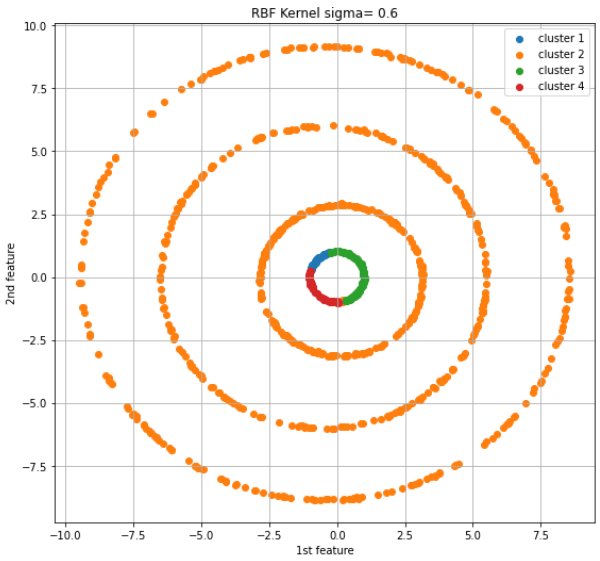

RBF Kernel sigma= 0.1
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature

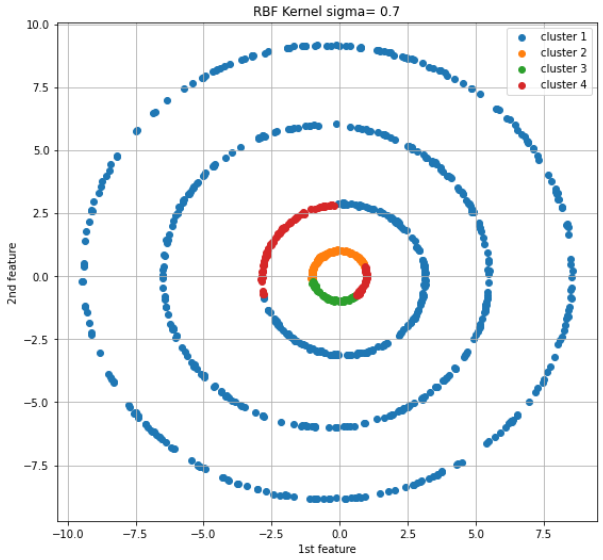

RBF Kernel sigma= 0.2
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature

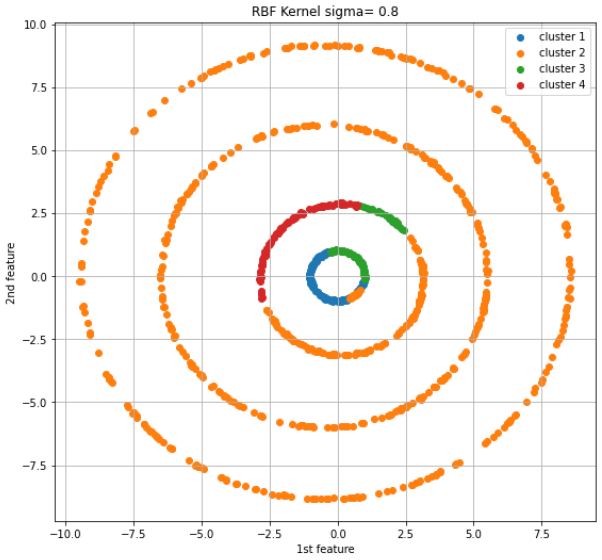

RBF Kernel sigma= 0.3
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature

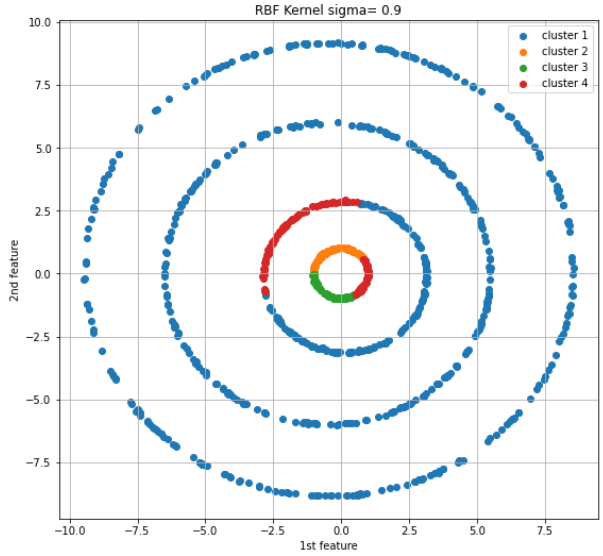

RBF Kernel sigma= 0.4
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature

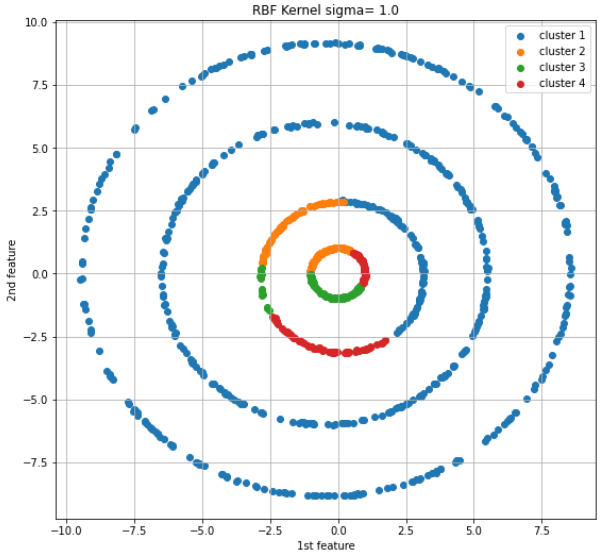

RBF Kernel sigma= 0.5
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5


RBF Kernel sigma= 0.6
cluster 1
cluster 2
cluster 3
cluster 4
2nd feature
1st feature
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5









Observation and analysis:

Here also we don't get the natural clusters and we may not get any voronoi region like in ques-2.py. 4 clusters are obtained with 4 different colors.

Choosing the best Kernel:

I am finding the Z or assignment matrix from rows of H matrix. Then I have found the mean of each cluster and then I have found error or objective function value for every kernel.The kernel which minimizes the objective function value is the best kernel.

- If I use polynomial kernel with degree -2 then error value is=27168.33
- If I use polynomial kernel with degree -3 then error value is=19785.17
- If I use RBF kernel with sigma-0.1 then error value is=31542.93
- If I use RBF kernel with sigma-0.2 then error value is=31536.87

- If I use RBF kernel with sigma-0.3 then error value is=31531.71
- If I use RBF kernel with sigma-0.4 then error value is=31501.60
- If I use RBF kernel with sigma-0.5 then error value is=31515.08
- If I use RBF kernel with sigma-0.6 then error value is=31478.99
- If I use RBF kernel with sigma-0.7 then error value is=31280.96
- If I use RBF kernel with sigma-0.8 then error value is=30748.33
- If I use RBF kernel with sigma-0.9 then error value is=31259.43
- If I use RBF kernel with sigma-1.0 then error value is=30791.55

Here error value is least in case of polynomial kernel with degree 3 . so it is the best kernel to choose in this case.