ASSIGNMENT 1

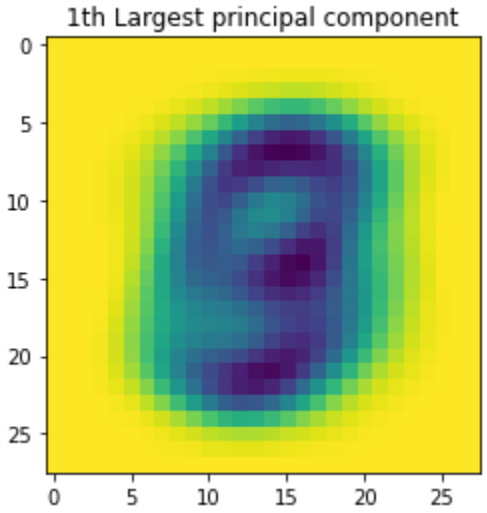
PRML

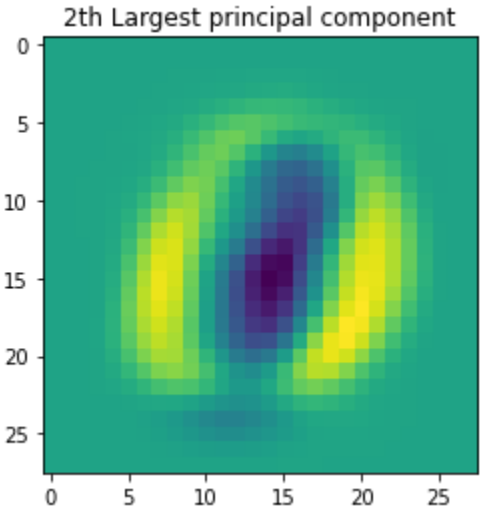
AKRANTH REDDY
ME20B100

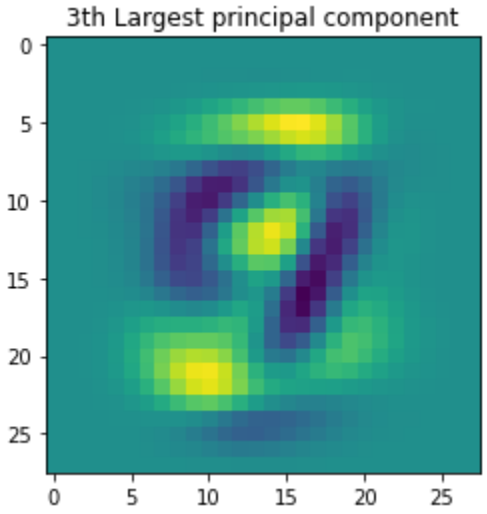
Question 1

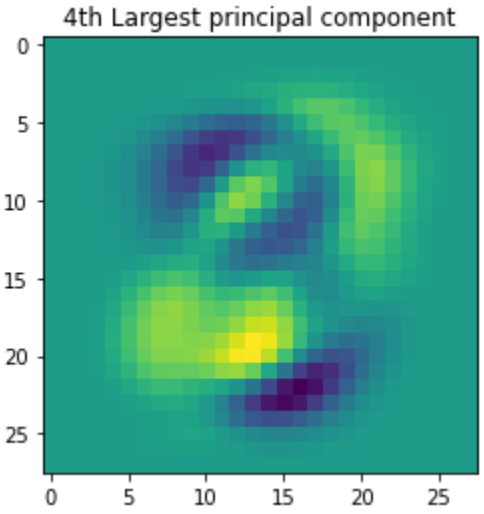
(i)

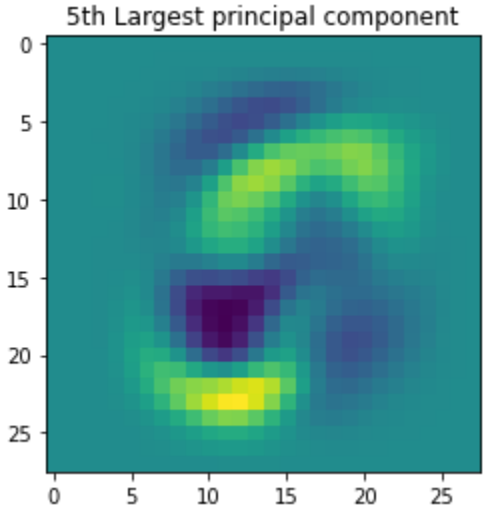
Images of principal components

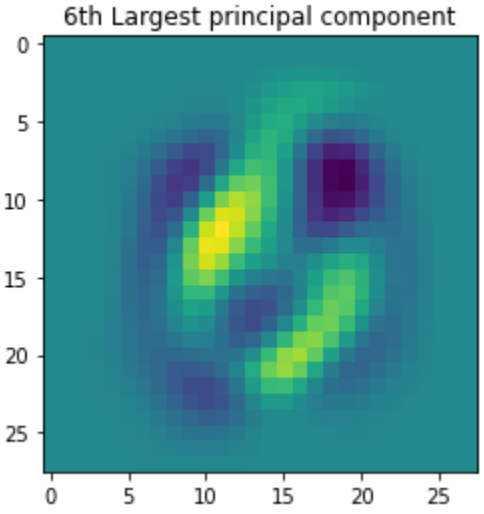


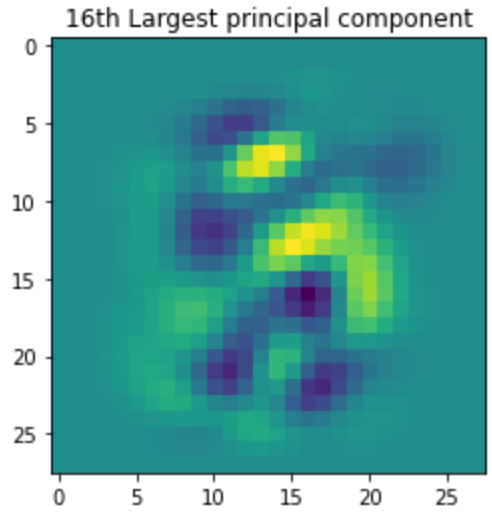


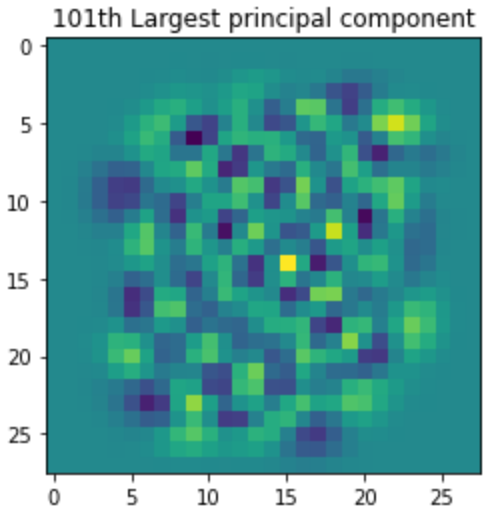


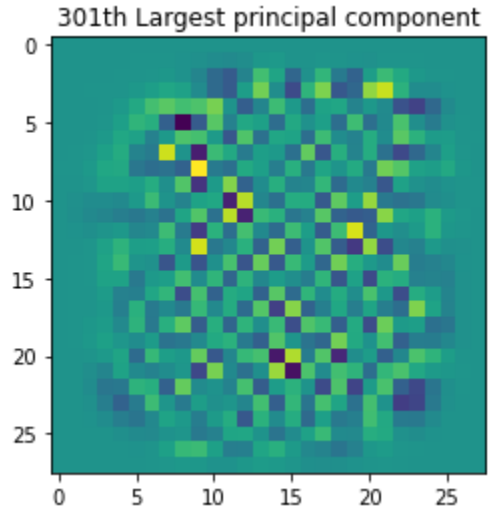


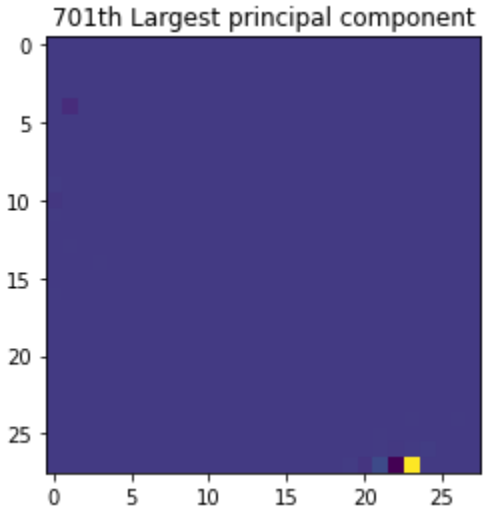




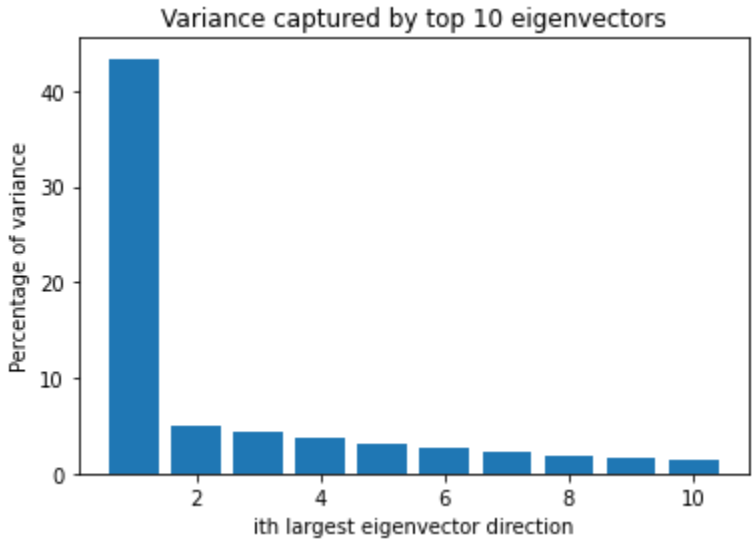


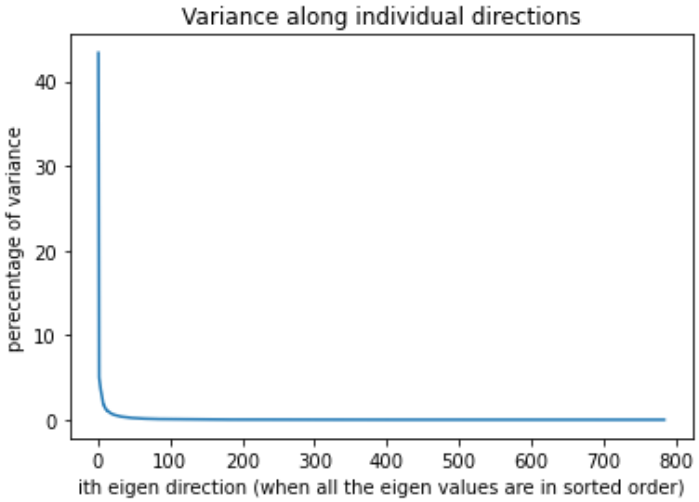


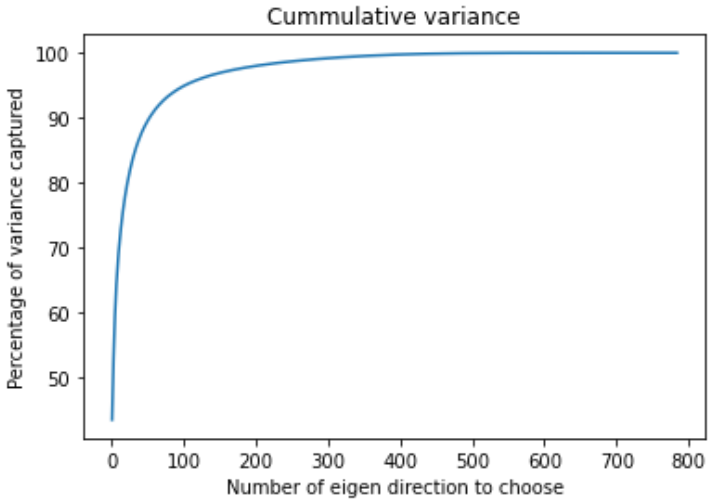




Variance explained by by principal components



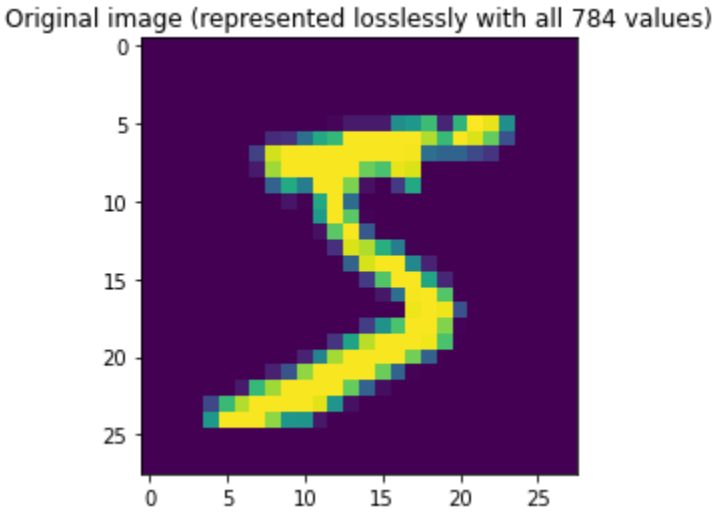




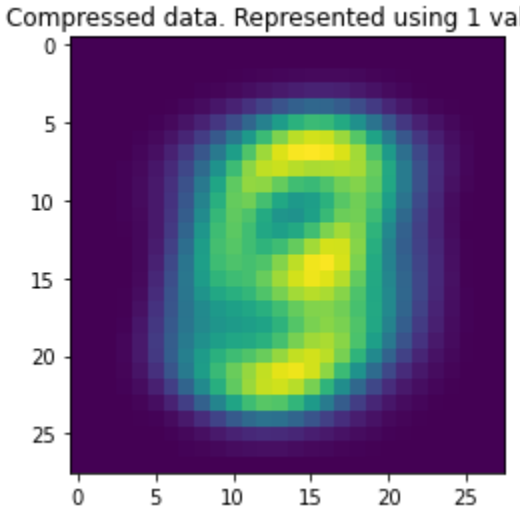
- The largest eigenvector direction (principal component) captures 43% of variance
- Top 10 principal component together captures 69.16% of variance
- Top 100 principal component together captures 94.87% of variance
- Break even point for capturing 95% of variance of 102 principal components.

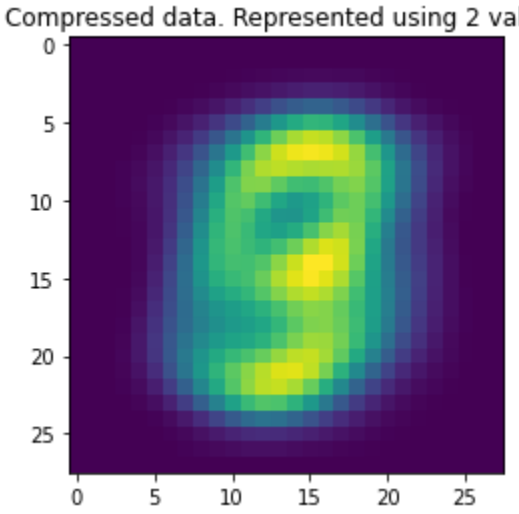
(ii)

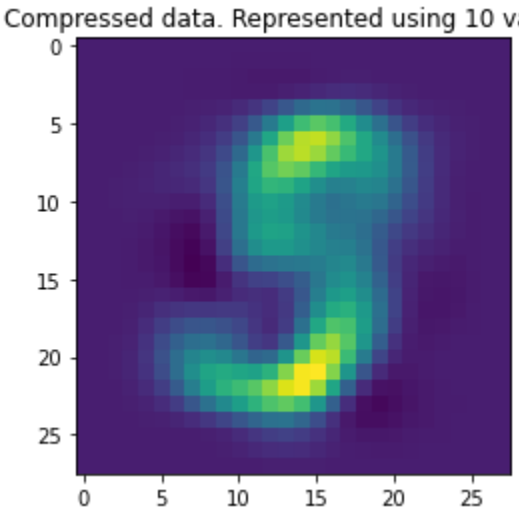
Now we try to represent ‘5’ in compressed form.

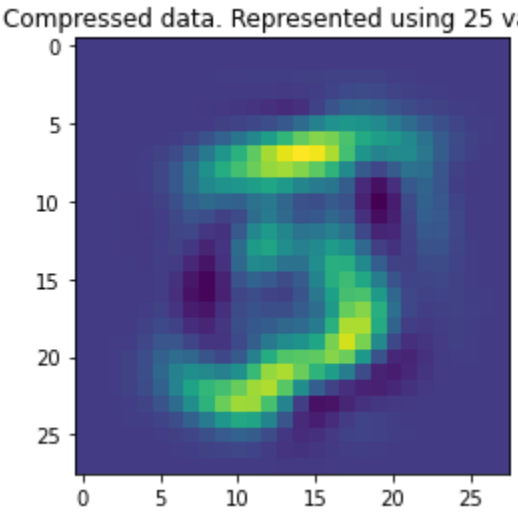


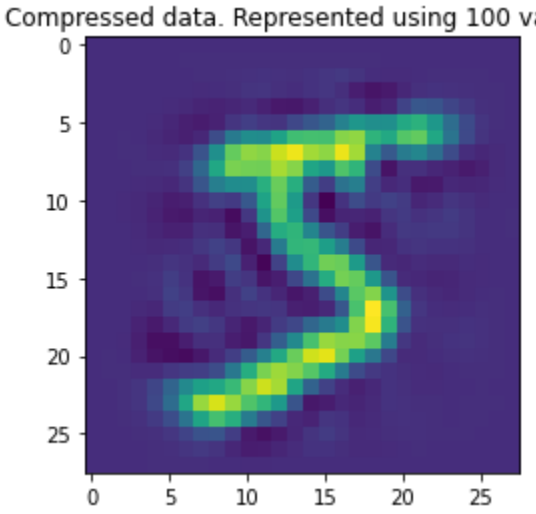
Compressed data

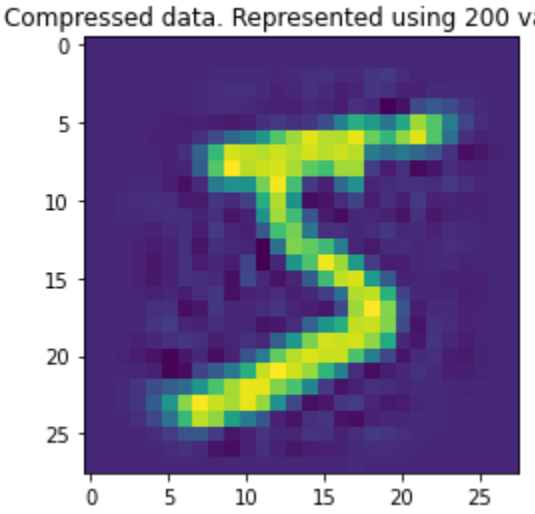


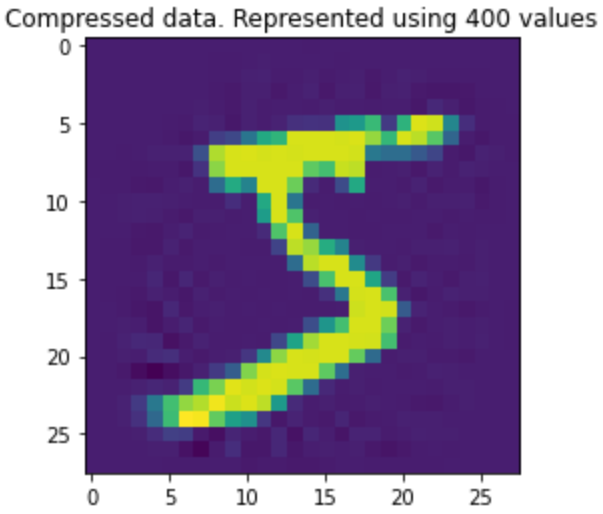


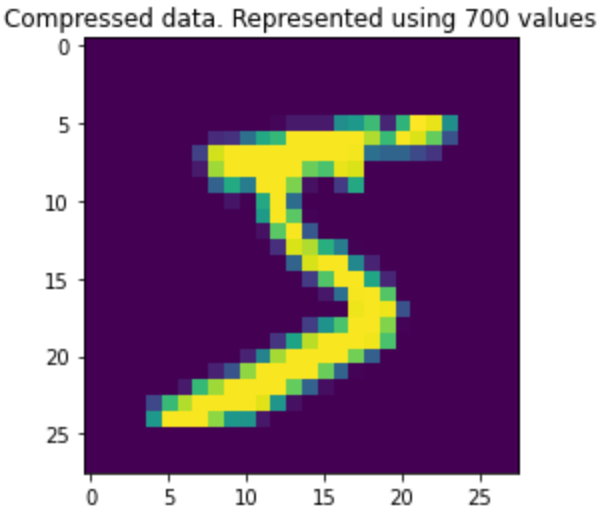








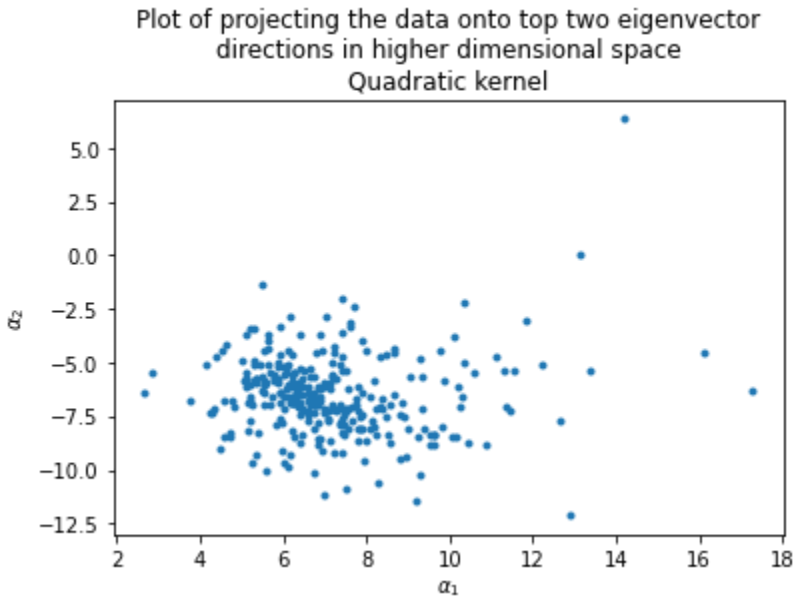


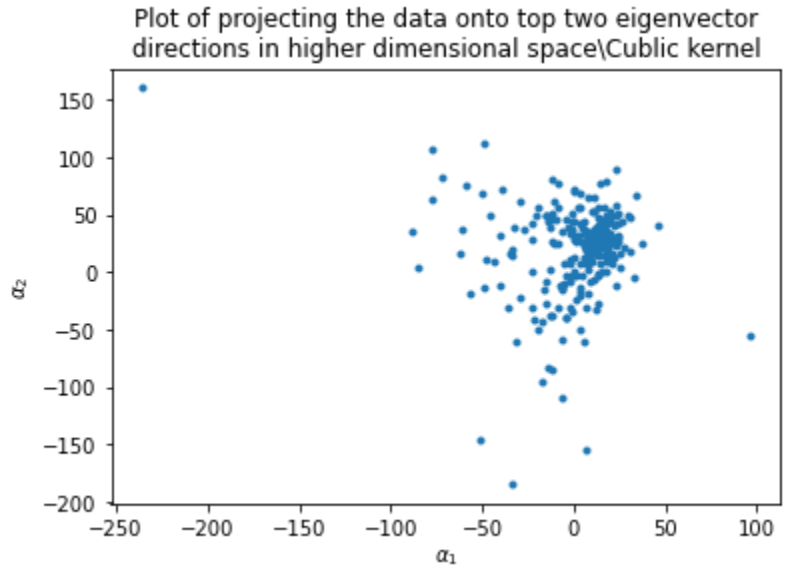


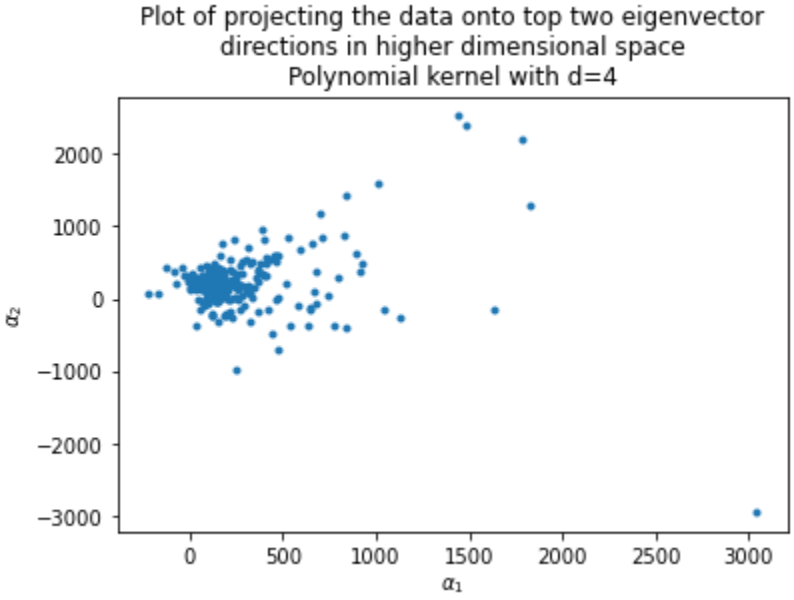
As we can see as more data points are used the better the variance of the original data is captured. But for typical downstream tasks it is enough if we can capture around 95% of variance of data. From graphs and code we can see using 102 principal components capture just close to 95%.

(iii)

Polynomial kernel.

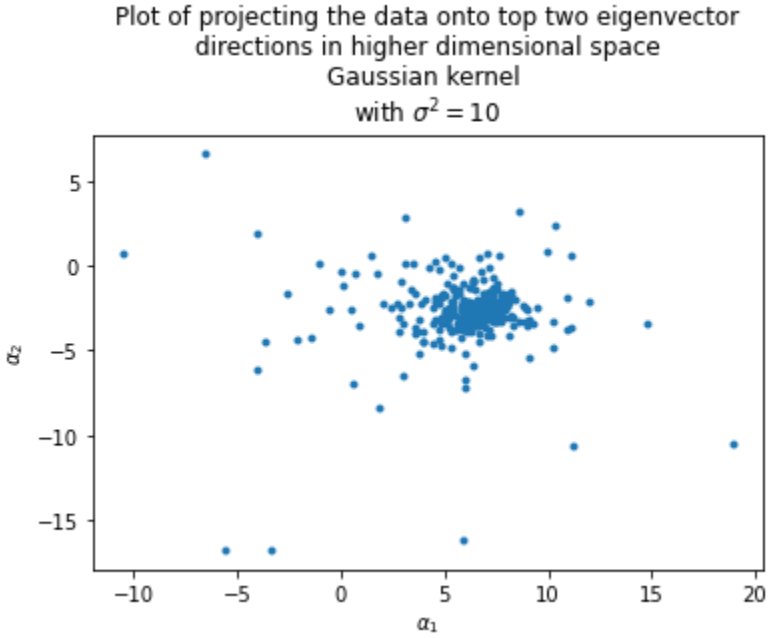


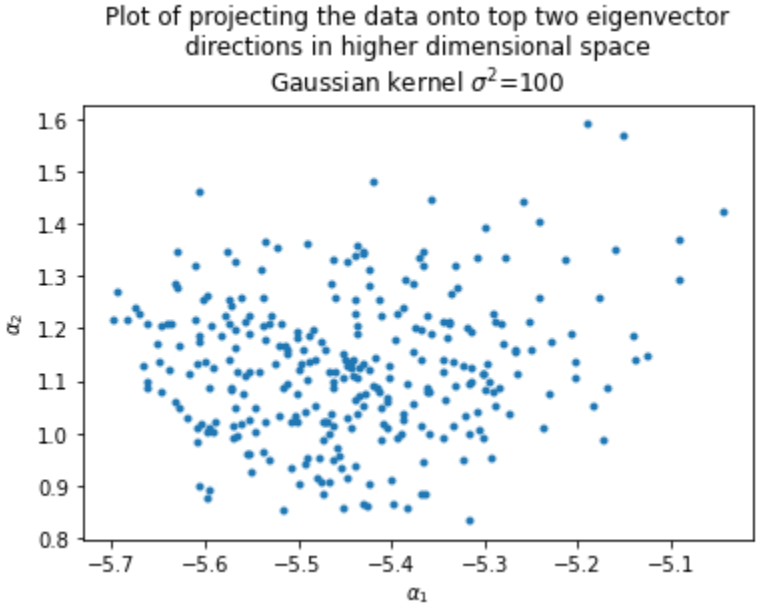

Plot of projecting the data onto top two eigenvector
directions in higher dimensional space\Cublic kernel
150
100
50
0
−50
−100
−150
−200
−250
−200
−150
−100
−50
0
50
100
α_2
α_1

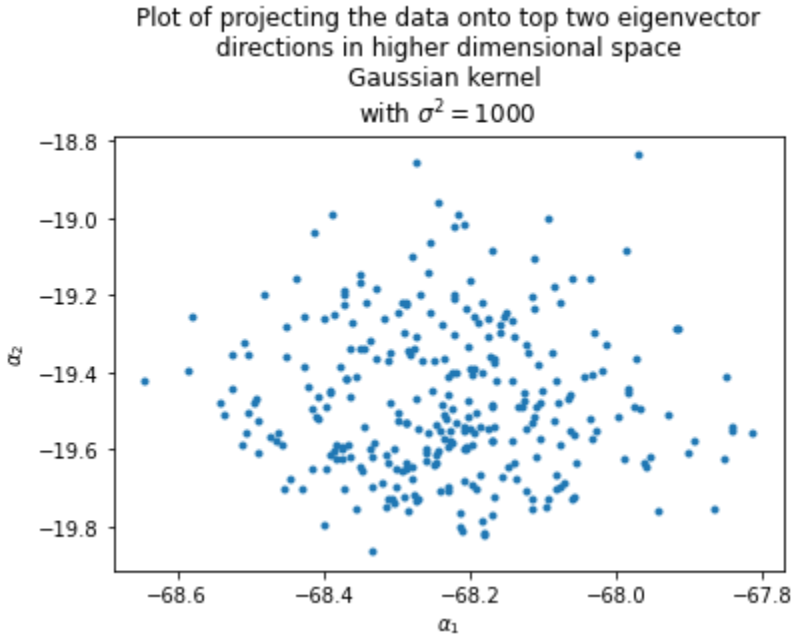

Plot of projecting the data onto top two eigenvector
directions in higher dimensional space
Polynomial kernel with d=4
2000
1000
0
−1000
−2000
−3000
0
500
1000
1500
2000
2500
3000
α_2
α_1

Gaussian Kernel

When a gaussian kernel is used with variance between 0.1 and 1 (as given in question) the compiler is not able to handle huge values generated. The values are crossing the limit 111,000 digits so I increased the variance and made plots to compare.







(iv)

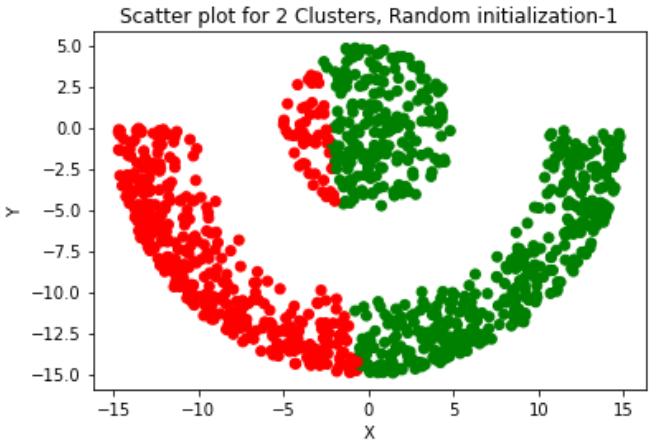
From observing the data we can see that for polynomial kernels all the data points are close, which is not favorable to us because if they are close we can't distinguish different clusters easily, but for gaussian kernels they are spread out.

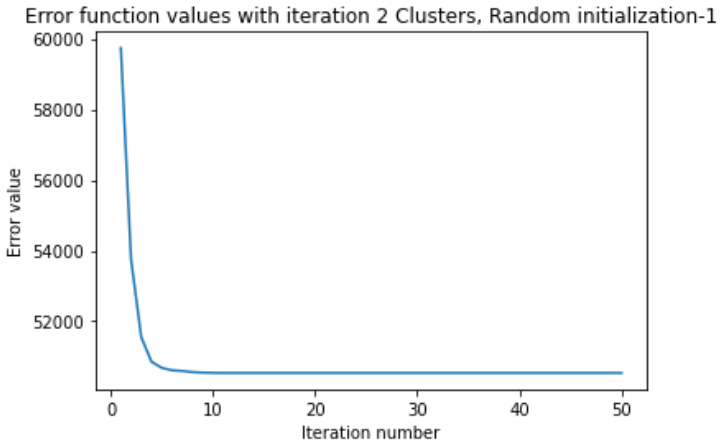
So a gaussian kernel is better in this case.

Question 2

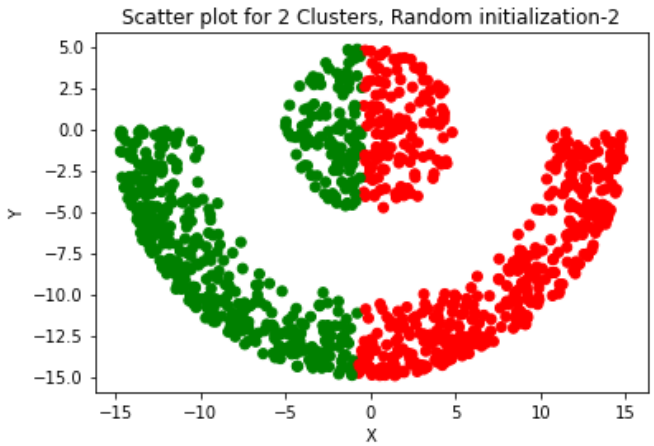
(i)

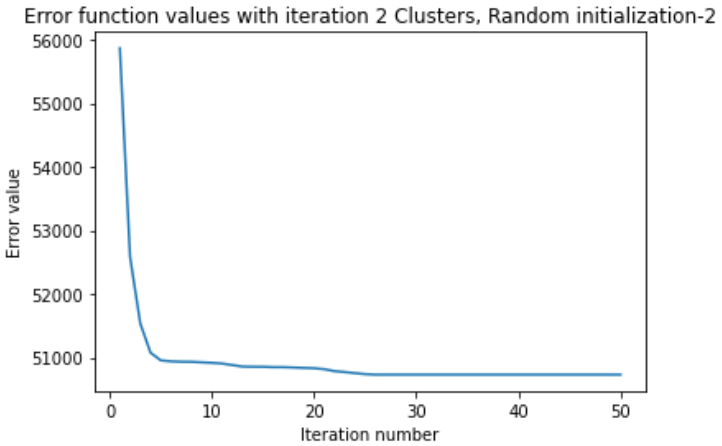
Random initialization - 1



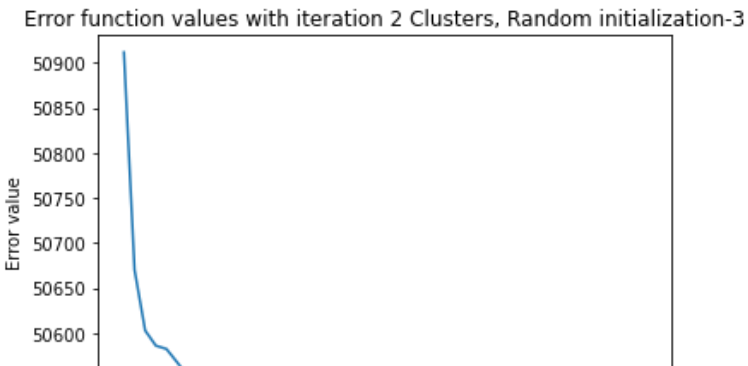


Random initialization - 2

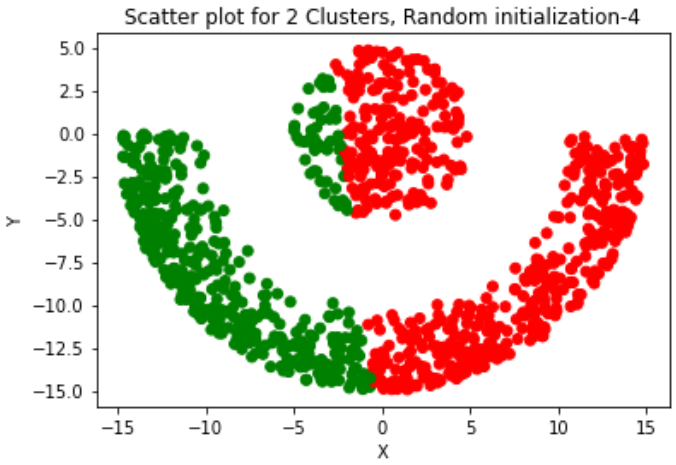


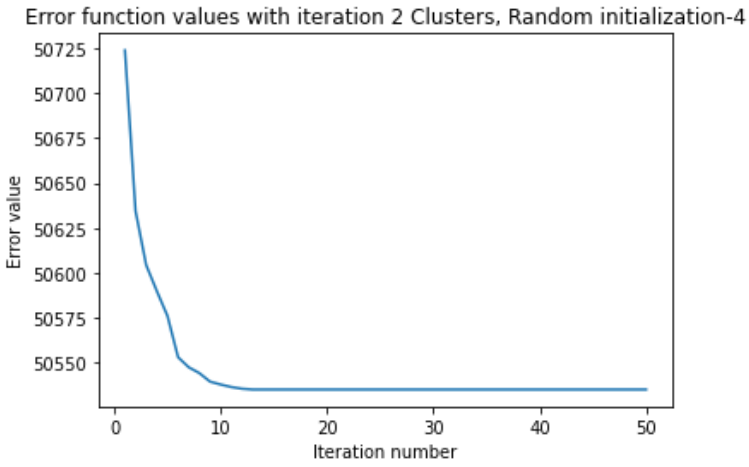


Random initialization -3

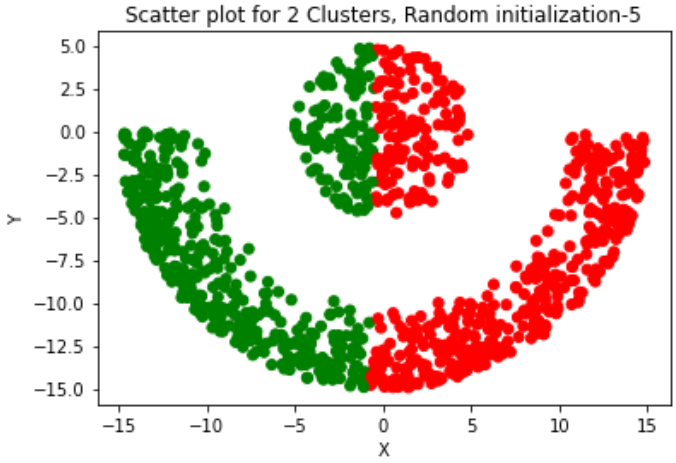


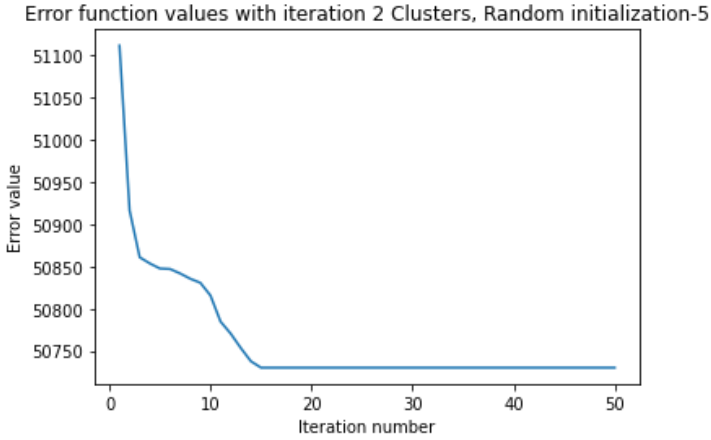
Random initialization -4





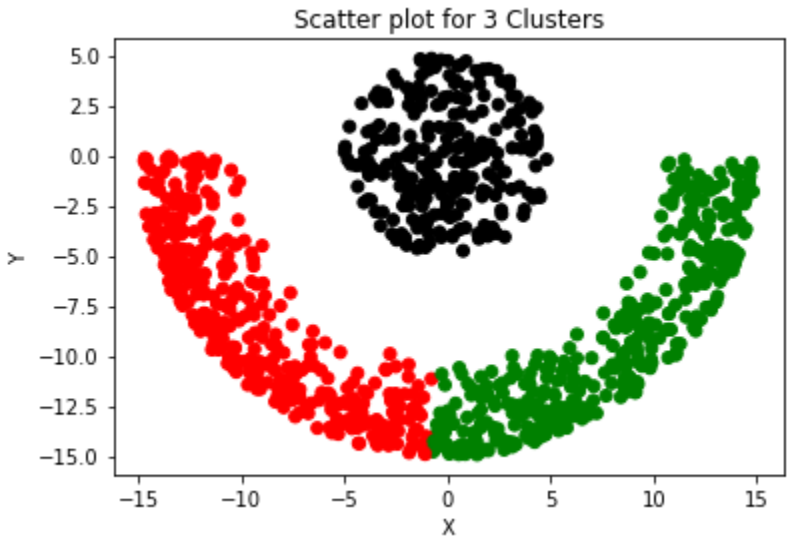
Random initialization -5

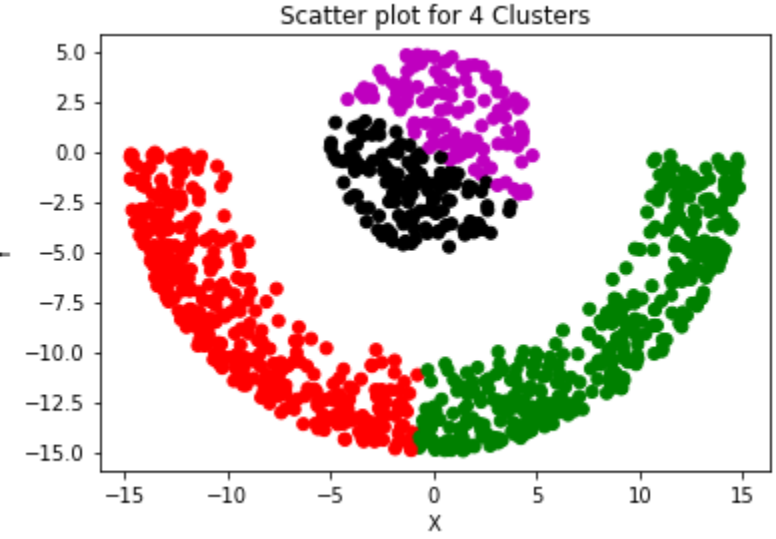


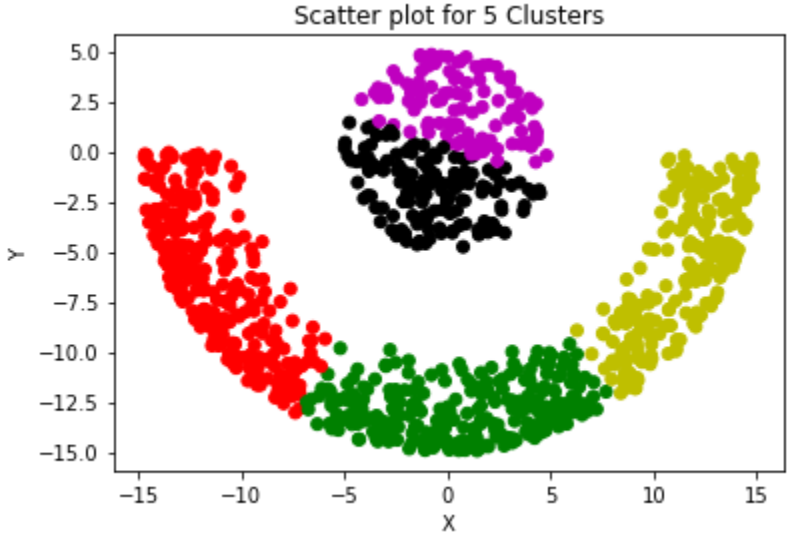


(ii)

Fix a random initialization and change values of k

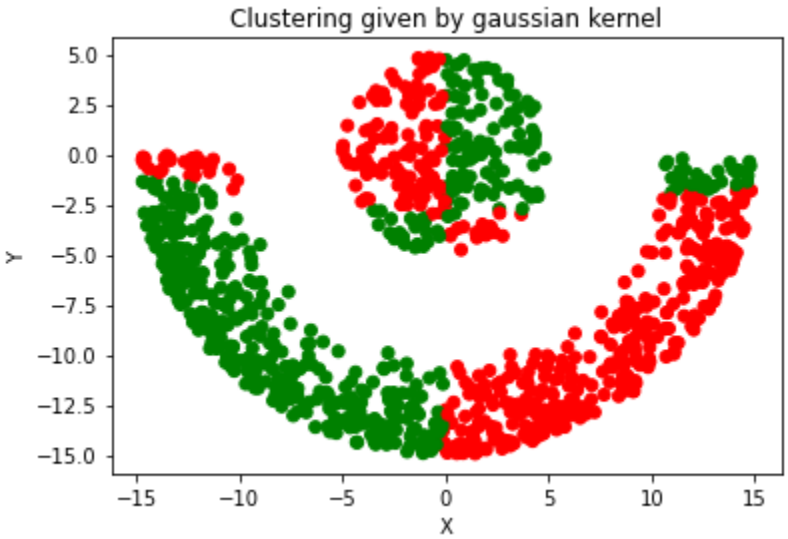




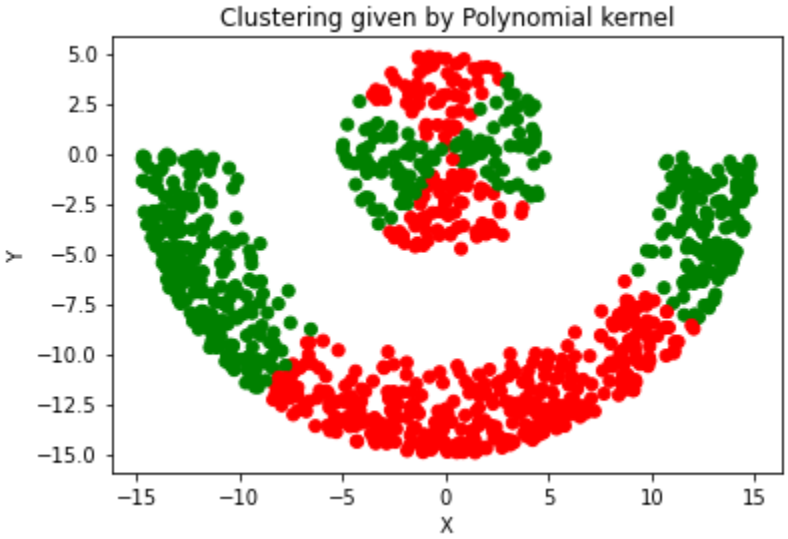


(iii)

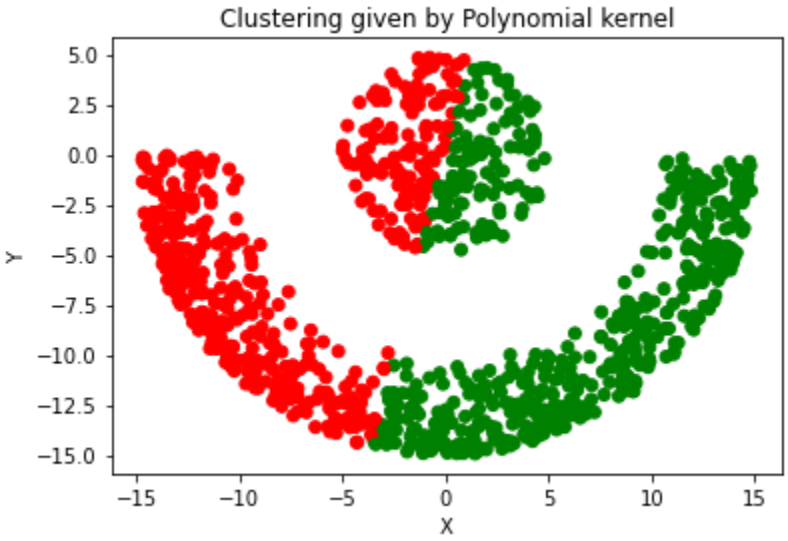
Gaussian kernel with variance = 10



Polynomial kernel of degree 2



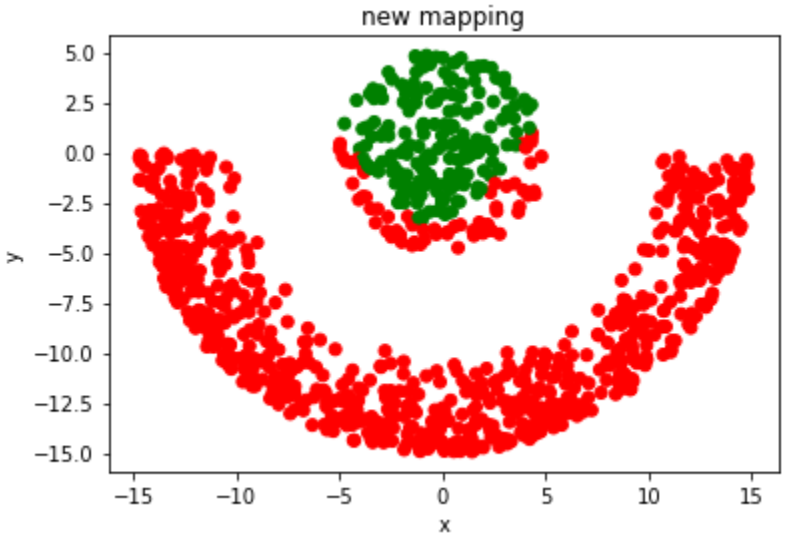
Polynomial kernel of degree 3



Ideally we want to have two clusters with a center circle in one cluster and remaining in another cluster. But these kernels are performing poorly. They are not doing what we want ideally, but they do capture the nonlinear parameters which otherwise might have missed out.

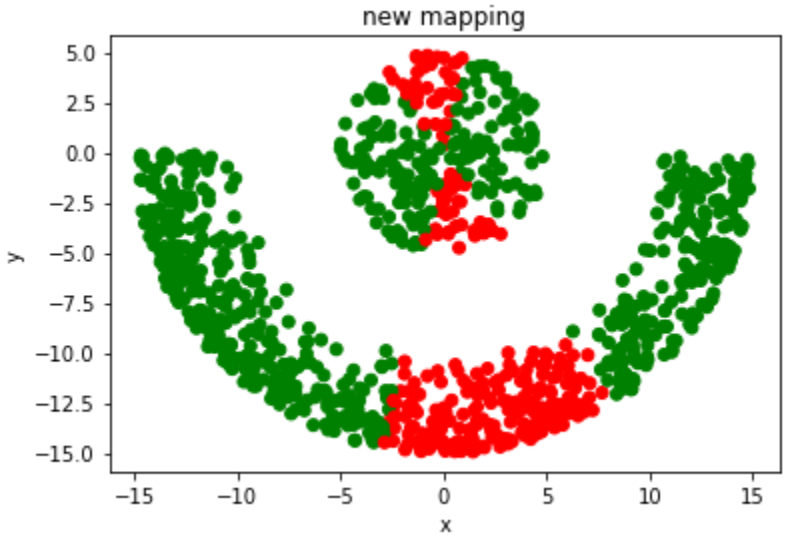
(iv)

Using other mapping. With gaussian kernel and variance 10.



This mapping is what we want, the center of data belonging to one cluster and the ring belonging to the other cluster.
So the new mapping function worked because We choose H and it so happens that the exact solution that we are looking for is far near to ZL^0.5 that's why we got a near/approx solution.

New mapping with polynomial kernel (quadratic)



This is clearly not helpful.