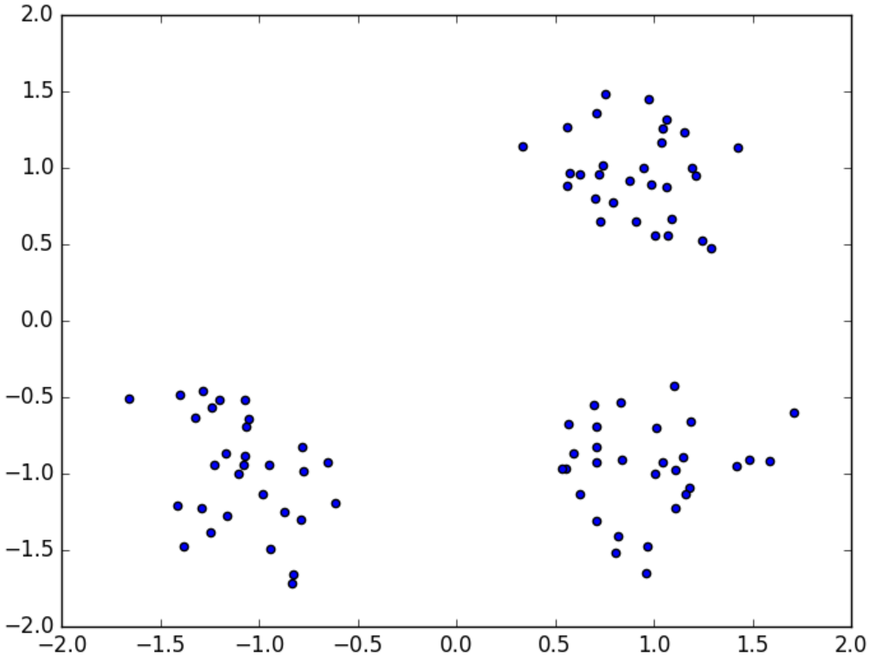
PART a)

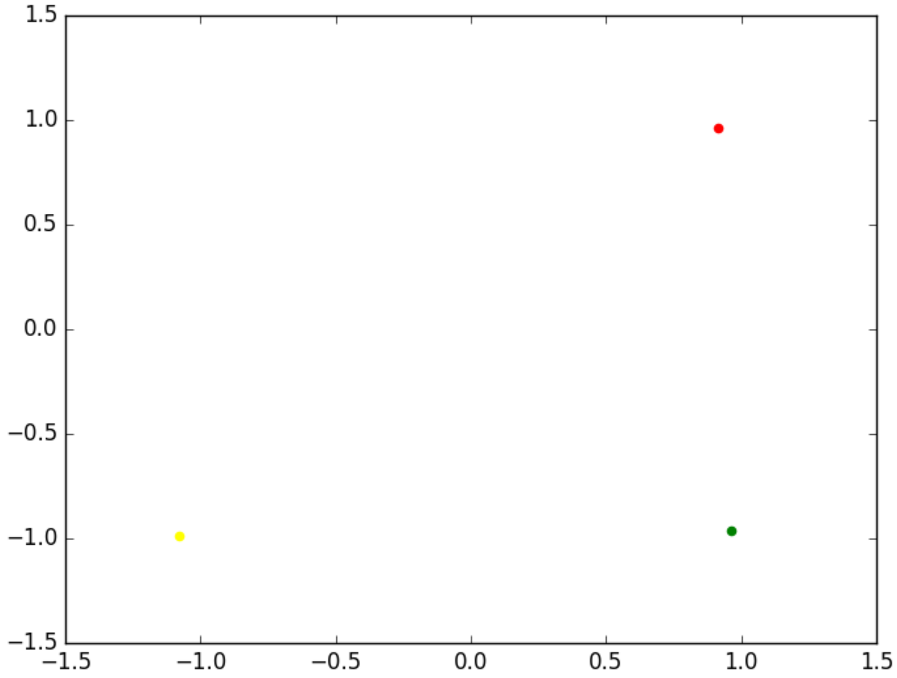
Implemented Agglomerative Clustering with following measures :

1 : Distance from mean (calculating distance form mean of cluster)
2 : Minimum Distance (calculate distance from all points, and take the minimum)

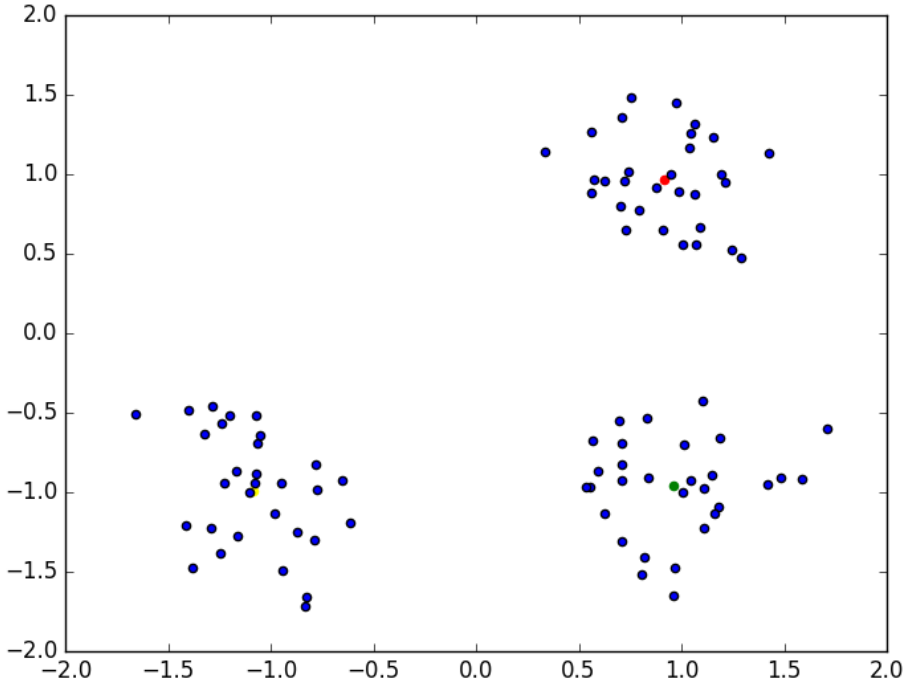
Following was the cluster that was built :



Now these were the centres that were detected :

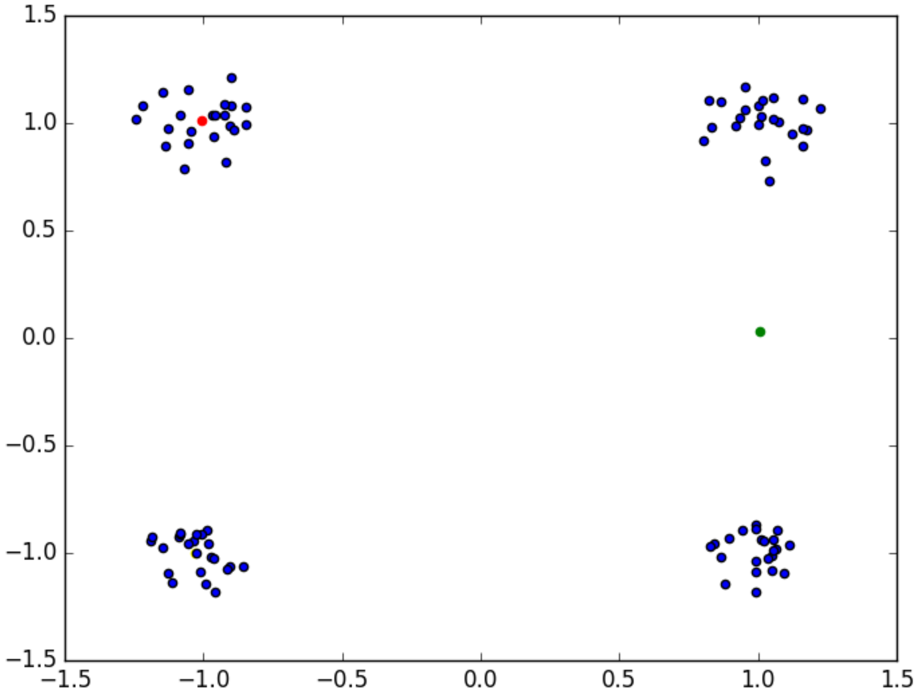


Plotting togethar both :



For both the methods used, the result was same.

---Now we used four clusters but printed only three centres :

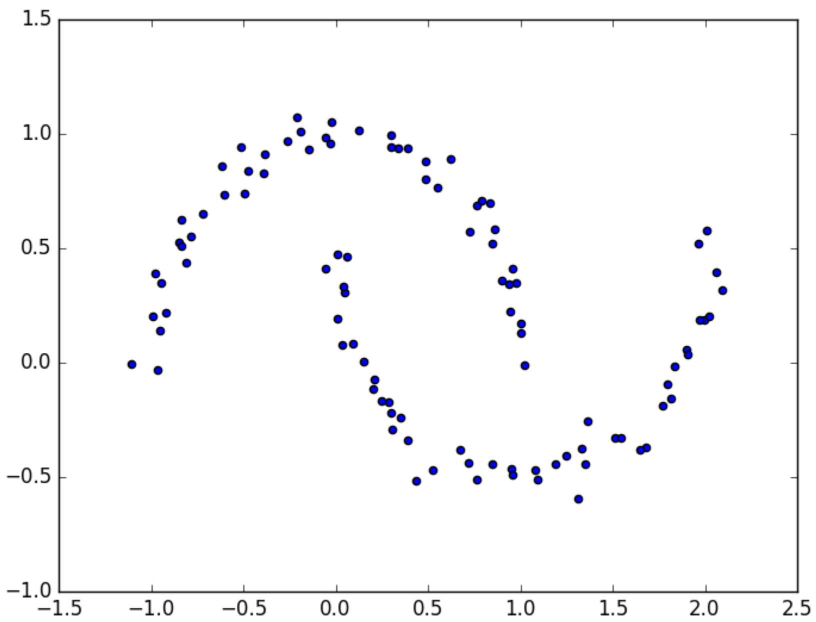


We can see 2 clusters have correct centres , but green is centre for both clusters.
If we go 1 step deeper in tree , we will get the correct four clusters.

PART b)

Spectral Clustering :

This is the cluster that we are seeking to seperate.



Used an RBF kernel to compute the similarity.

From this similarity matrix , we build the 'WEIGHT MATRIX'

If two points are far , similairity is less , hence weight assigned is less
If two points are close, similairity is high , hence weight assigned is more.

In this way , we can DIRECTLY use Rbf kernel to compute 'W' with a slight modification :
When similarity is too less (below a threshold) , we set weight as 0 (no connection in graph)

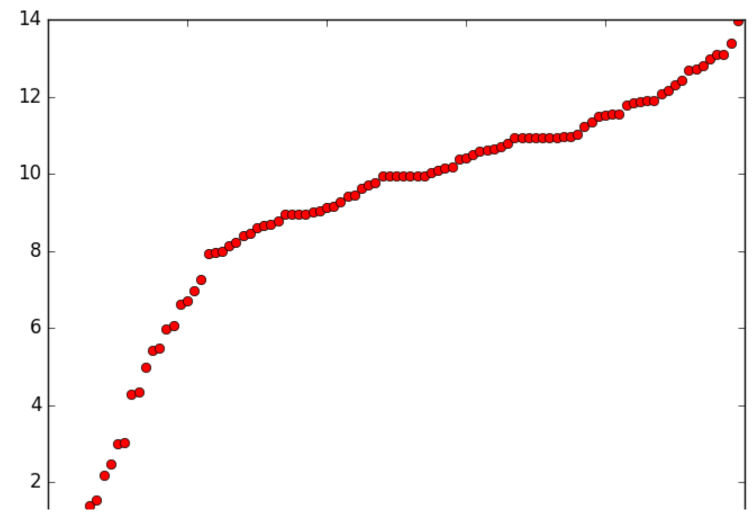
The Threshold used was 0.980 (meaning all points whose similarity was less than this, do not have a edge between them).

Thus we assign weights to the Undirected Graph .

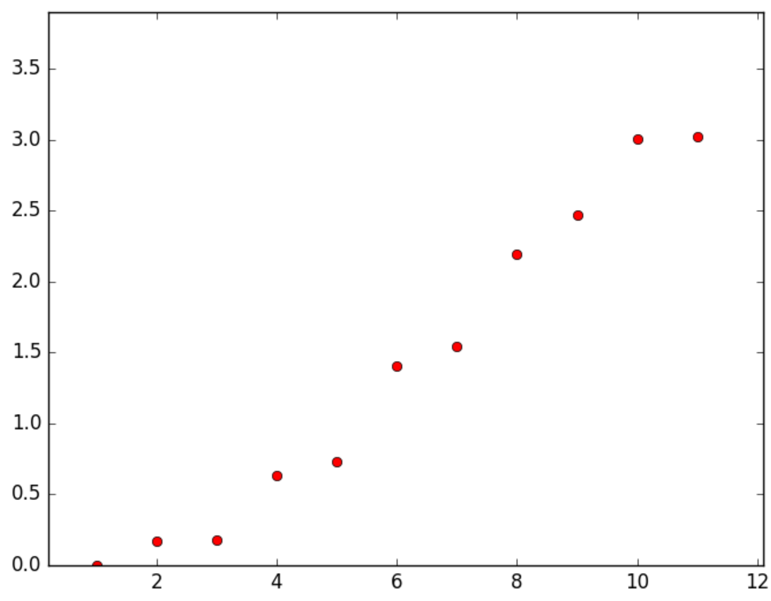
Then we build a Simple Laplacian Matrix as : L = D-W

W: Weight Matrix
D : Diagonal Matrix having Degree of Vertices as elements.

Then we find the Eign Values :
They are as :

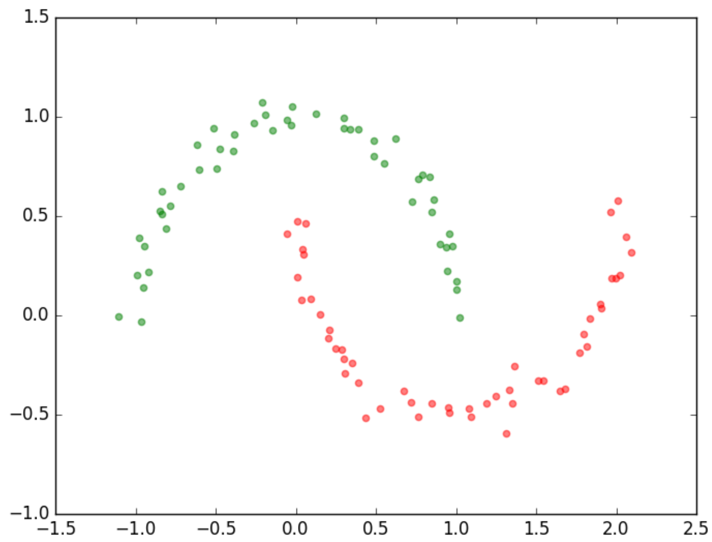


Looking more closely to the initial values :

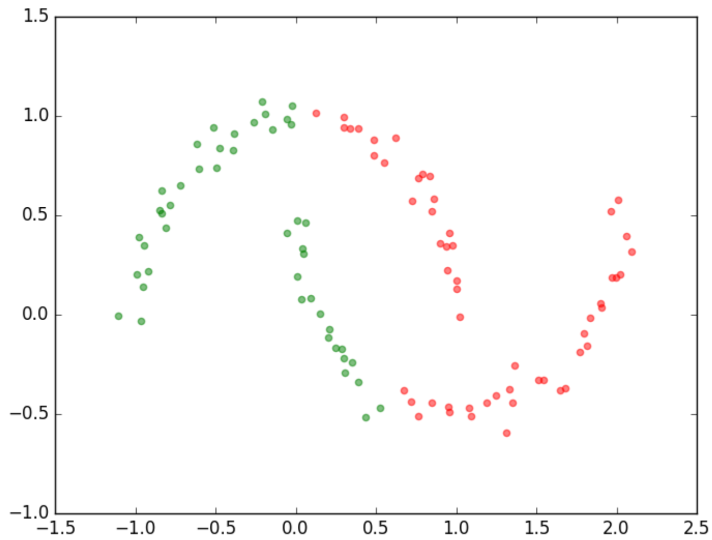


Here we can clearly see the **EIGNGAP.**

Then we use Second Eign Vecotor and get the following result :

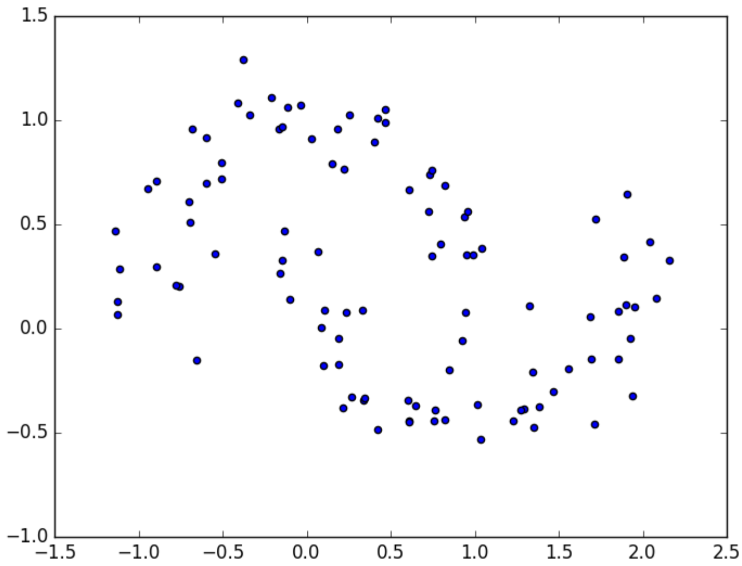


Then using vetor 1 to 4 we get :

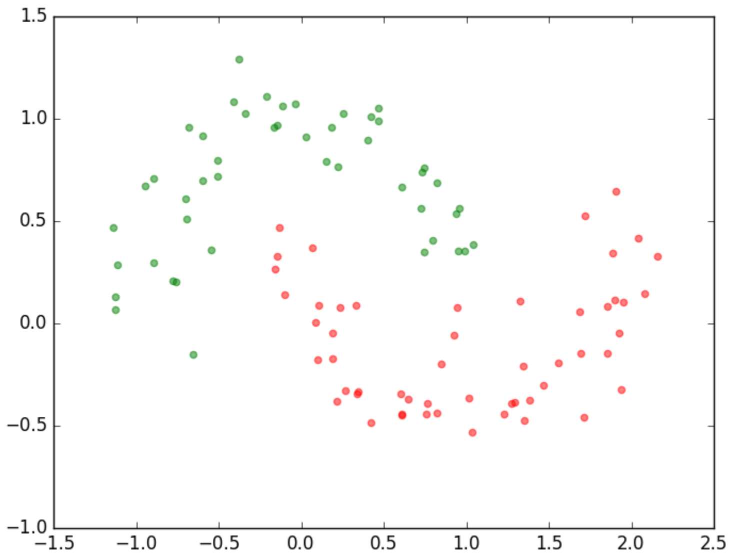


Clearly this is a poor result. So just taking first vector yeilds pretty good clusters.

Taking a very Noisy Cluster :



The model still performs very well .



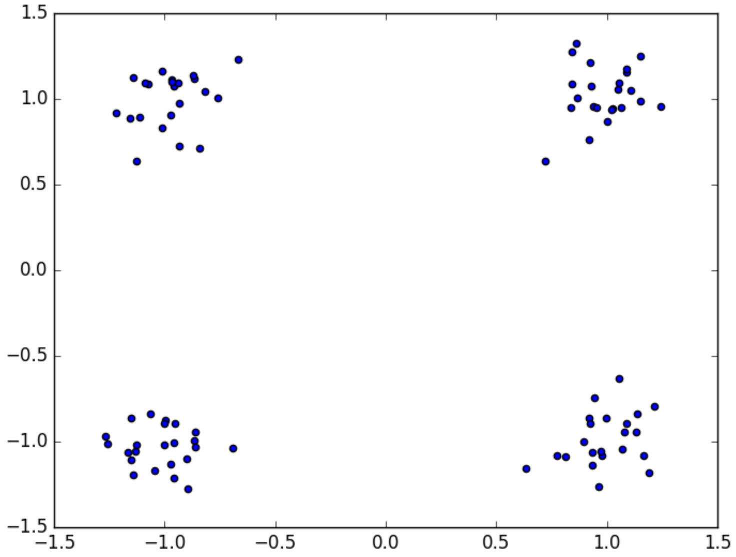
In this case , we can clearly see the power of Spectral Clustering.
The Half Moons are perfectly clustered, even for ambiguous places.

Also **implemented** simple K means algorithm to test the results.

PART c)

RBF Kernel :

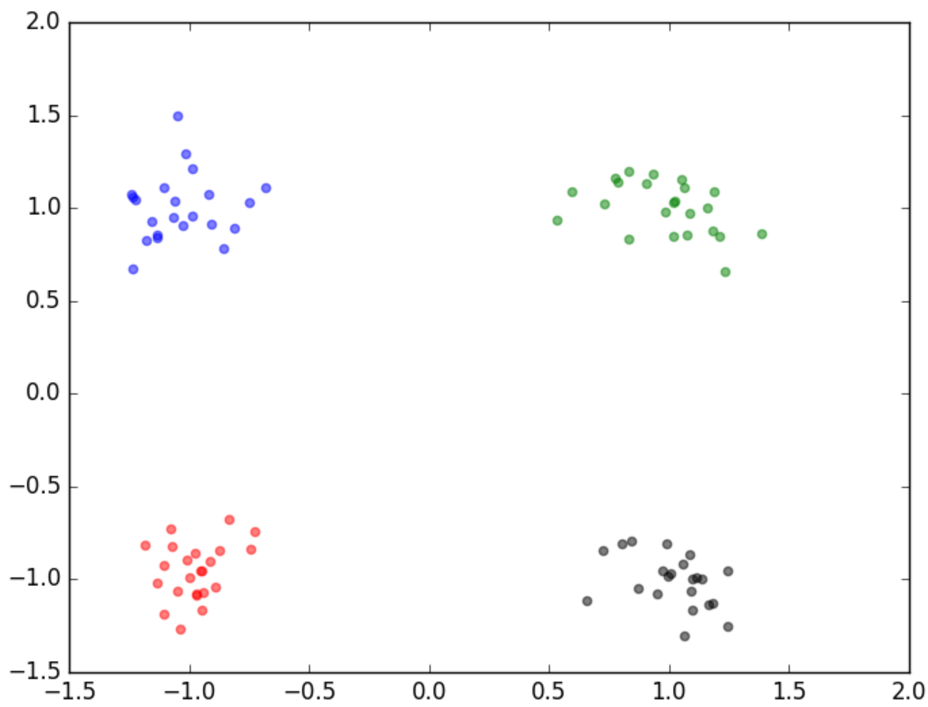
Following was the cluster on which the program was executed :



We had to set the value of K = 4.
Initial Means were choosen as :

```
random.seed()
indexes = random.sample(range(0, len(df)), k)
means = df.ix[indexes]
```

We got following clustering with 10 iterations

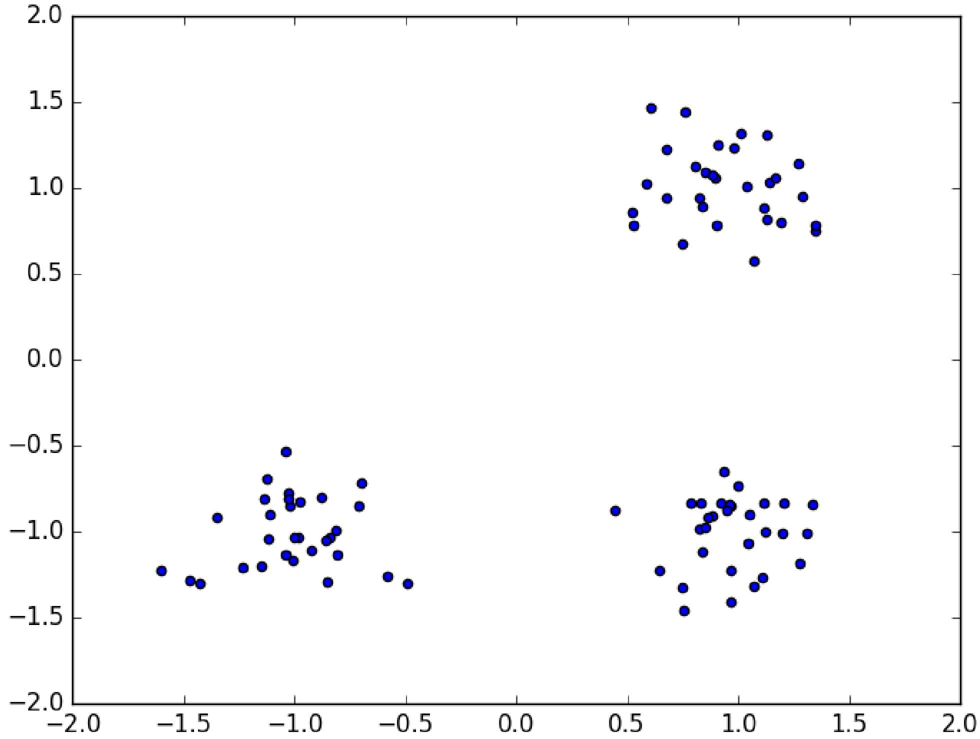


FOR POLYNOMIAL KERNEL :

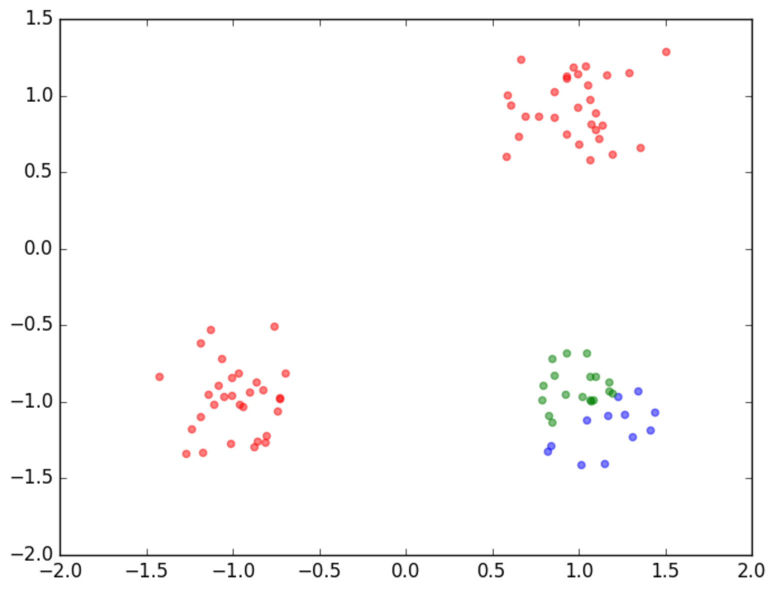
K(i,j) = (1 + x(i).T . x(j))^p
Used a Polynomial Kernel of Degree 2

```
kk = 1+np.dot(data[i],data[j].T)
kk=math.pow(kk,2)
```

Now we Cluster this Data :



With 4 iterations :



With 10 iterations :