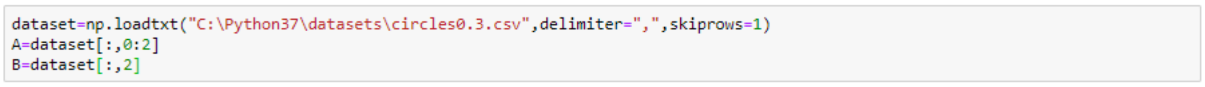
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Q1. Download the following datasets: circles0.3, moons1, spiral1, and half kernel.

A1)

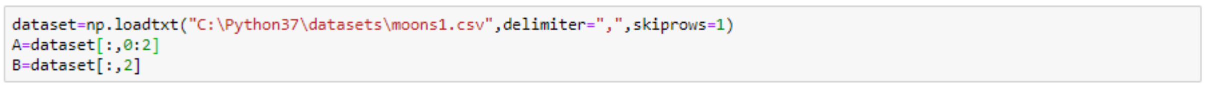
The following datasets have been downloaded from the resources which are: - circles0.3, moons1, spiral1 and halfkernel.
The following can be read in the Jupyter notebook using the code: -

Dataset: - Circles0.3



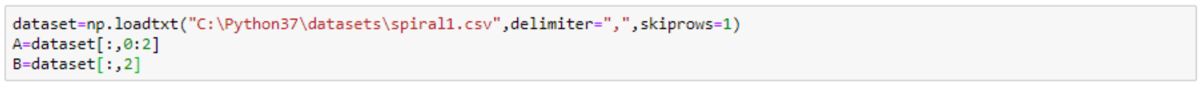
```
dataset=np.loadtxt("C:\Python37\datasets\circles0.3.csv",delimiter=",",skiprows=1)
A=dataset[:,0:2]
B=dataset[:,2]
```

Dataset: - Moons1



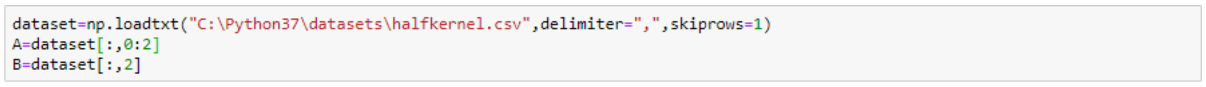
```
dataset=np.loadtxt("C:\Python37\datasets\moons1.csv",delimiter=",",skiprows=1)
A=dataset[:,0:2]
B=dataset[:,2]
```

Dataset: - Spiral1



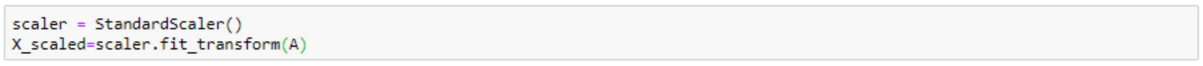
```
dataset=np.loadtxt("C:\Python37\datasets\spiral1.csv",delimiter=",",skiprows=1)
A=dataset[:,0:2]
B=dataset[:,2]
```

Dataset: - Half Kernel



```
dataset=np.loadtxt("C:\Python37\datasets\halfkernel.csv",delimiter=",",skiprows=1)
A=dataset[:,0:2]
B=dataset[:,2]
```

Now all the datasets are connected to the jupyter notebook and hence we can begin with the operation of standardizing the dataset.



```
scaler = StandardScaler()
X_scaled=scaler.fit_transform(A)
```

Similar standardizing statement or code is applied on all the other datasets **depending upon the clustering it is either applied or removed.**

Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

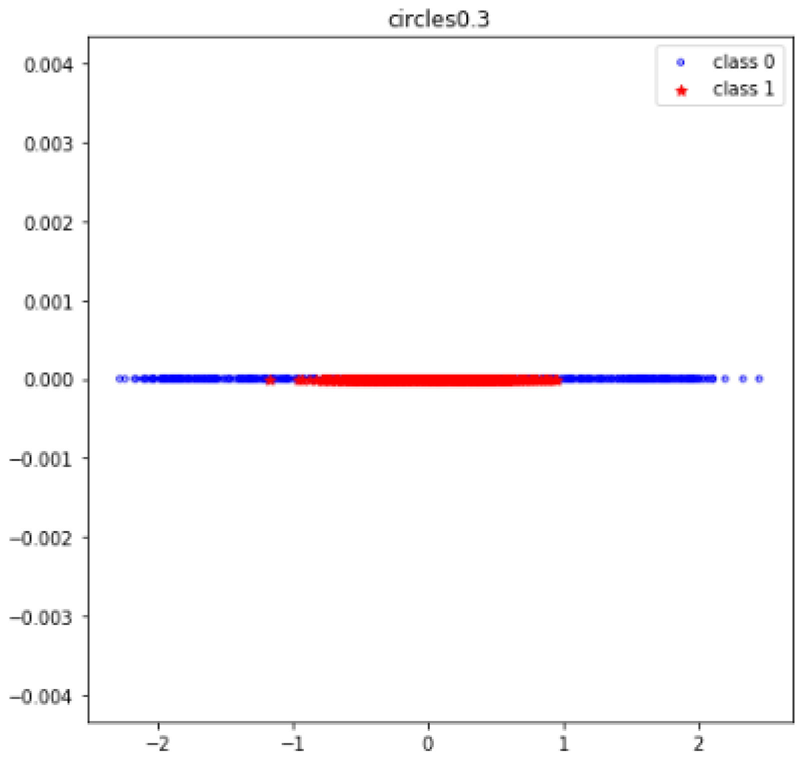
Q2&3. Run PCA and kernel PCA on all downloaded datasets, and project the data onto the one-dimensional space. Then, apply k-Means to the resulting data. Make sure you obtain a good clustering (marks will be deducted if reduction/clustering is not good).

Plot the samples of all datasets (in separate plots) as points in the original 2D space, and
on the projected space (one-dimensional space), using a different color and point shape
for each class. Highlight the resulting clusters in the one-dimensional space.

A2)

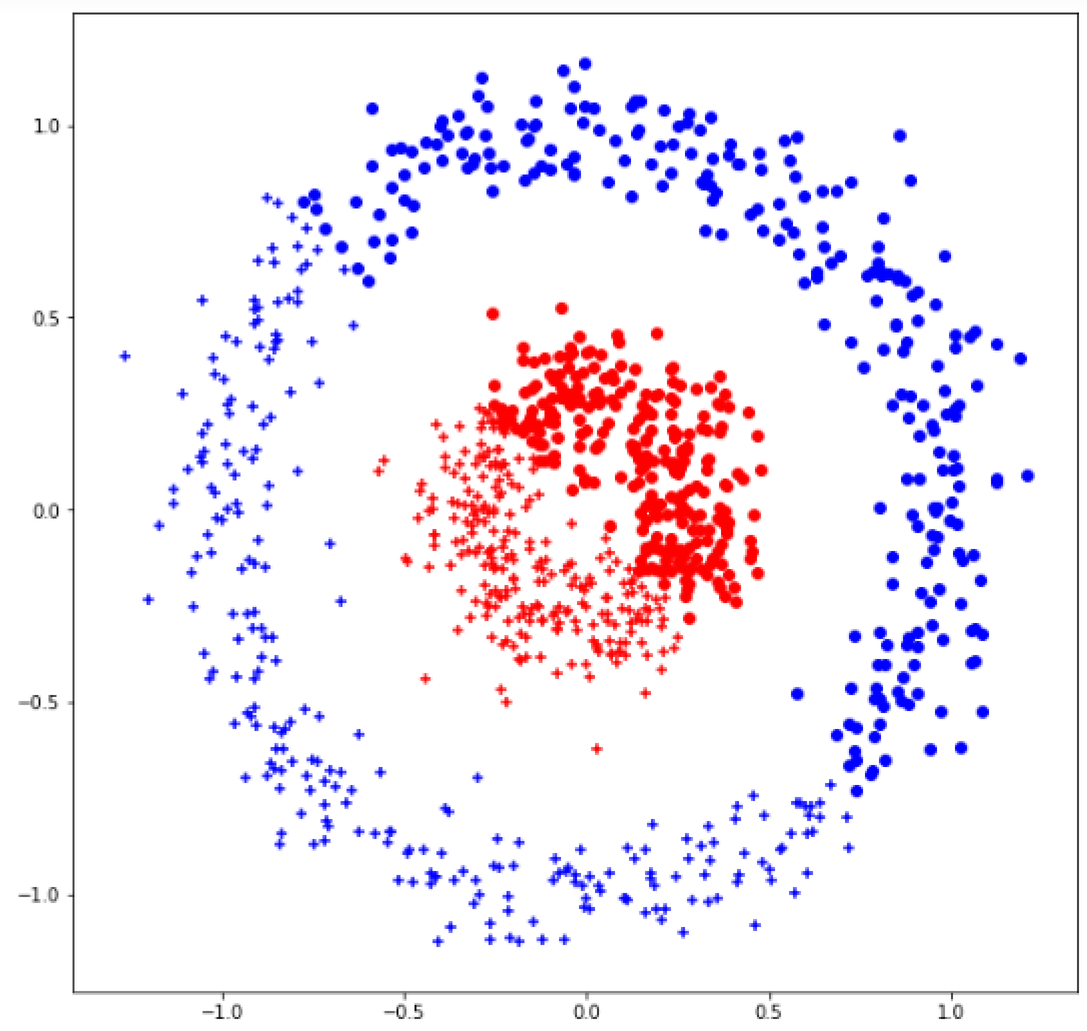
Dataset: - Circles 0.3

PCA in 1D



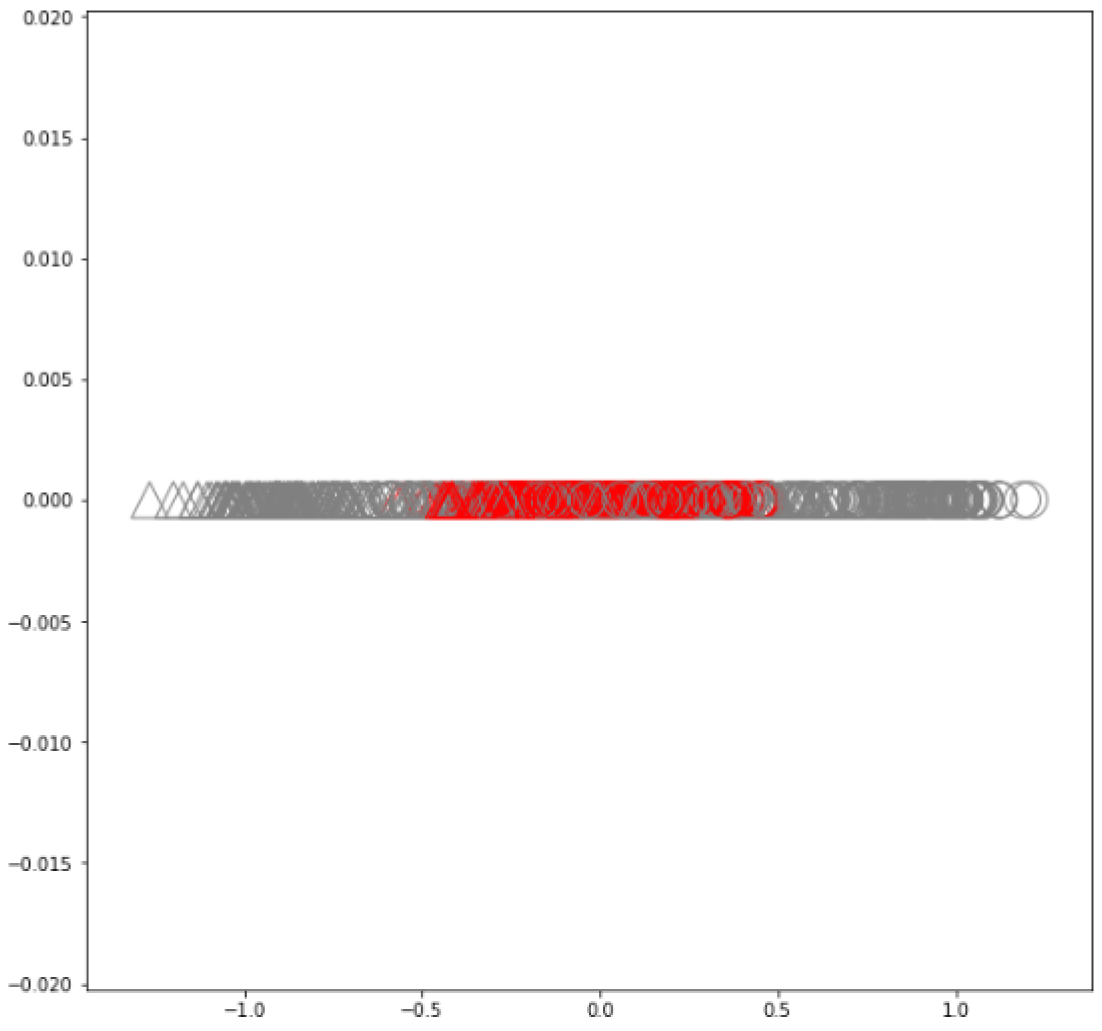
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying K Means on PCA



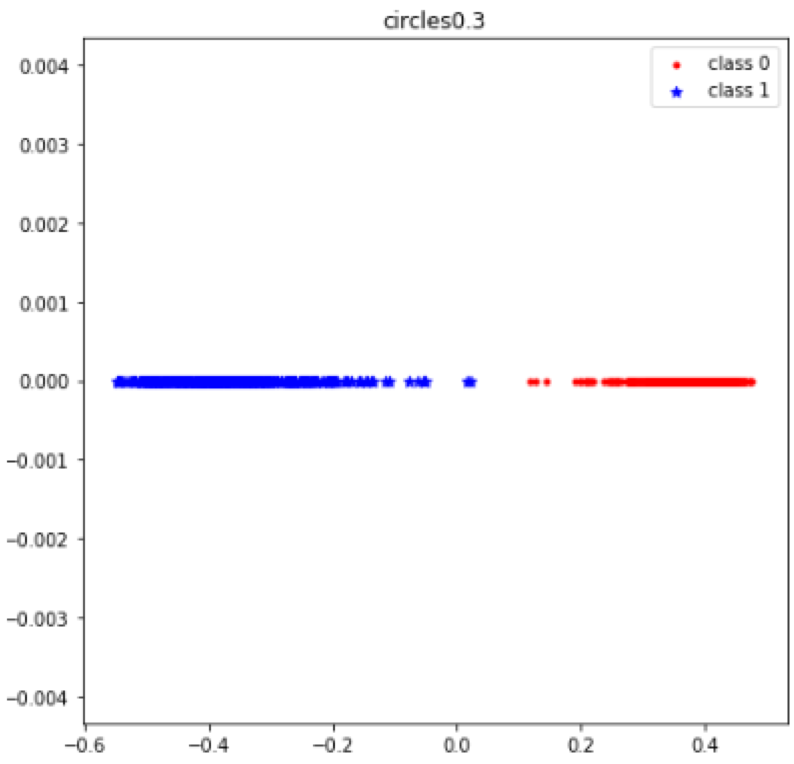
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans 1D



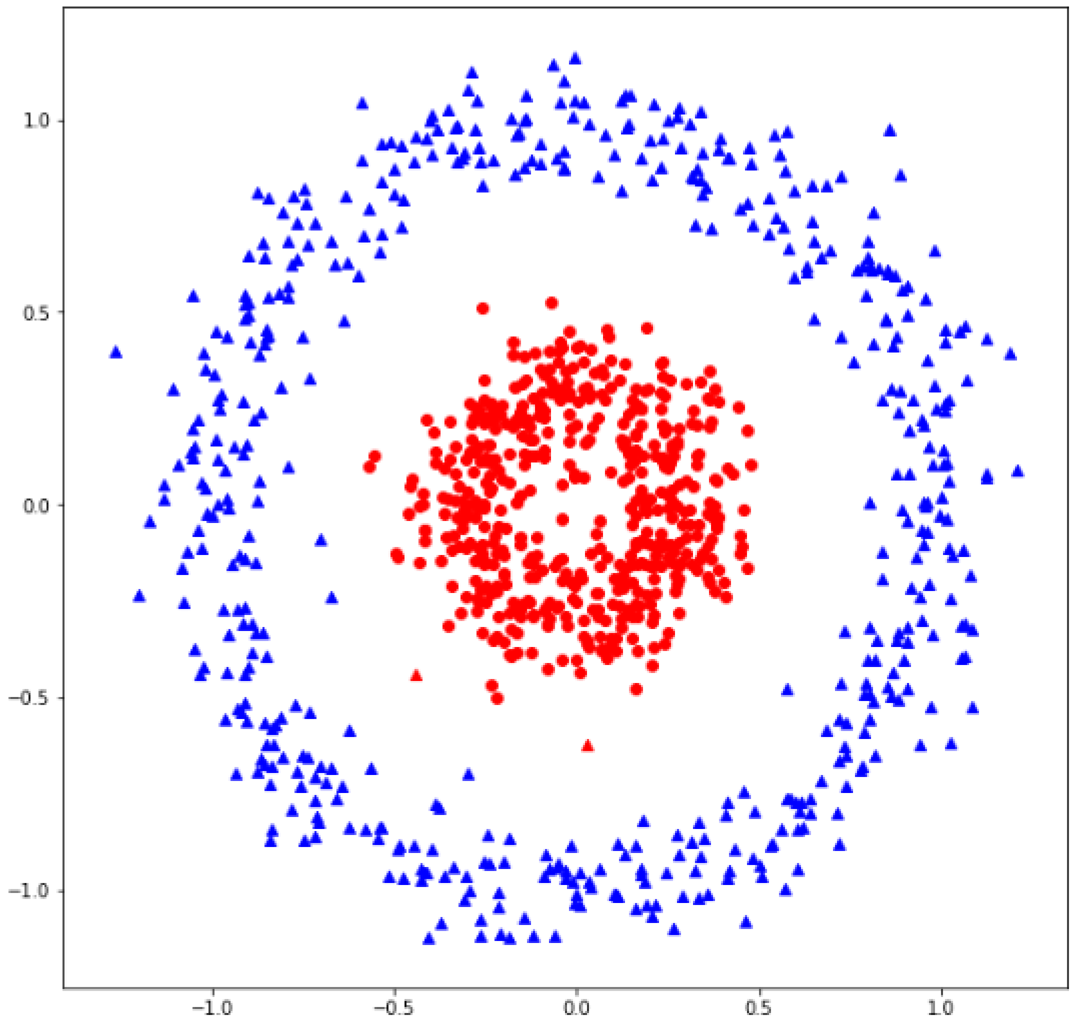
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

KPCA in 1D



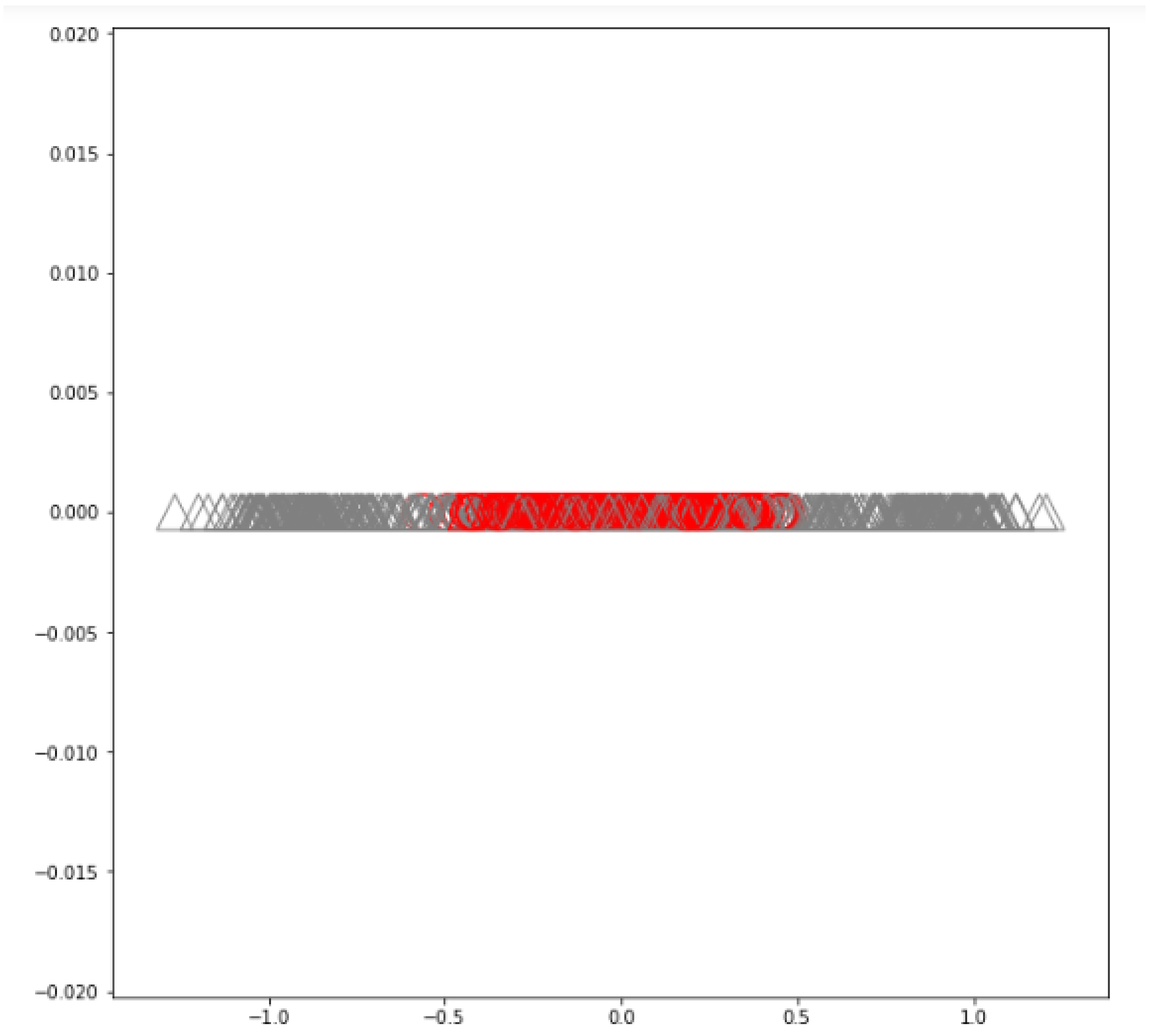
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans on KPCA



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

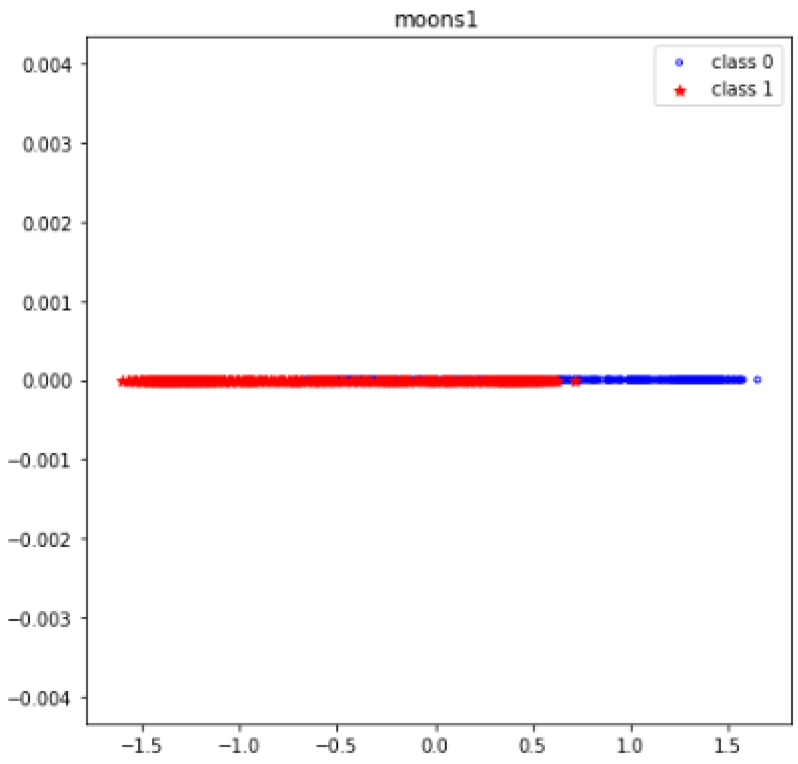
K means in 1-D



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

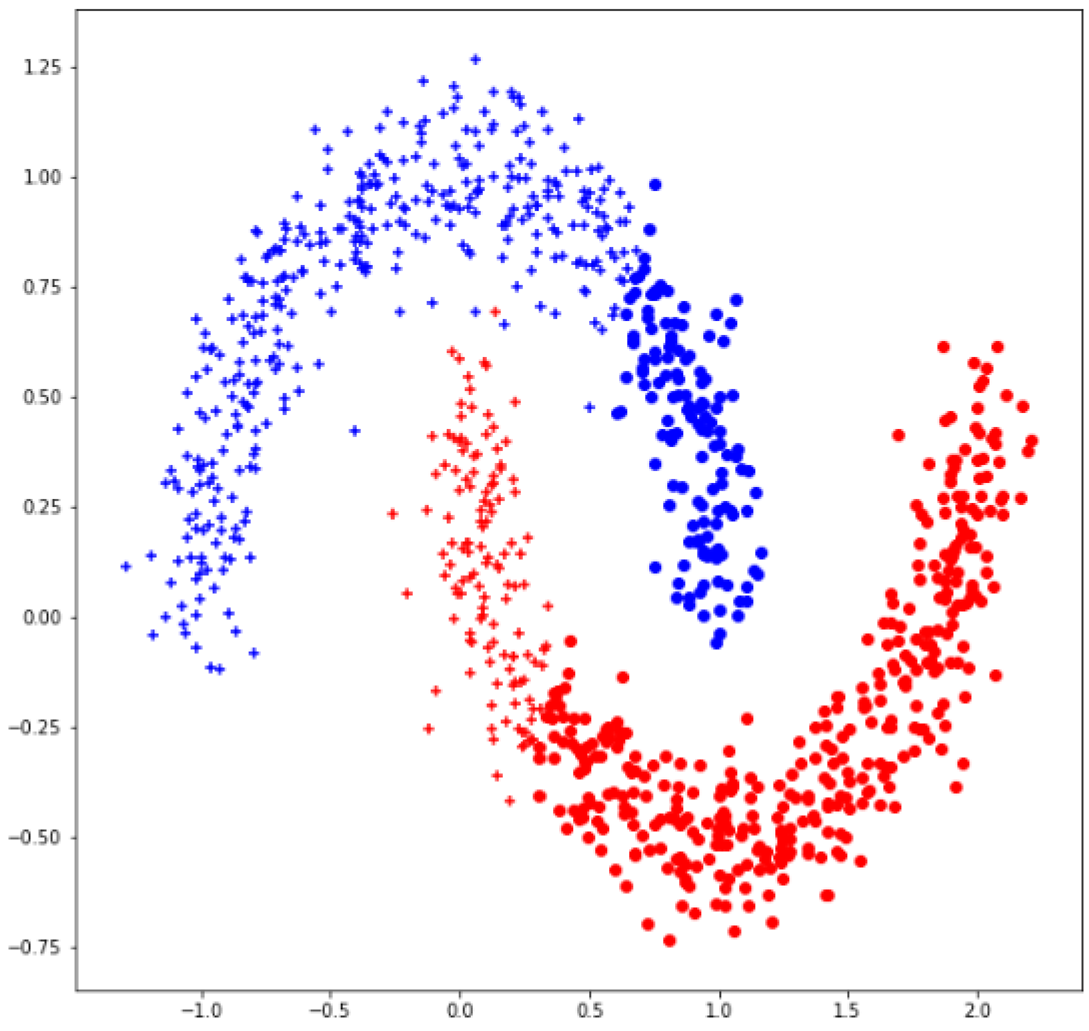
Dataset: - Moons1

Appling PCA in 1D



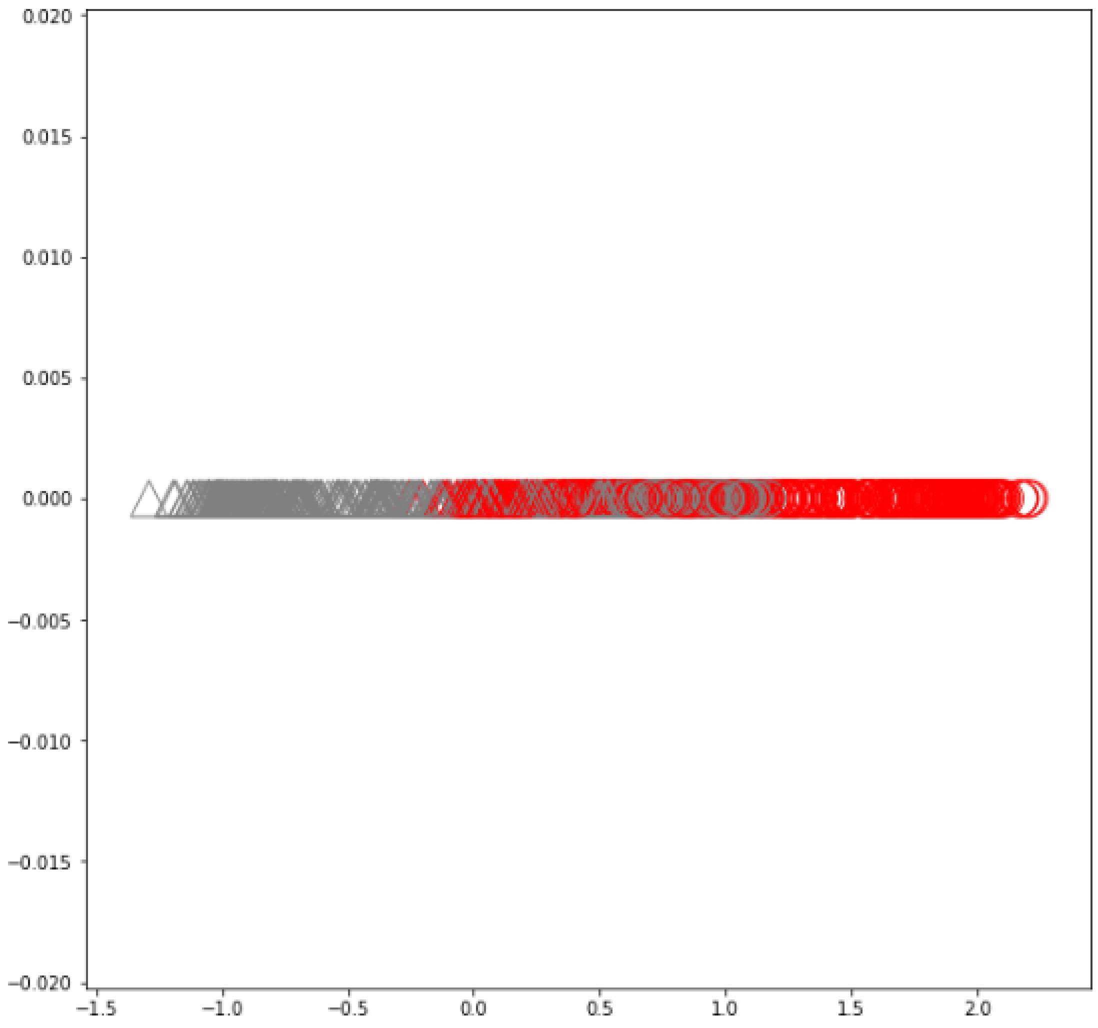
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans on PCA



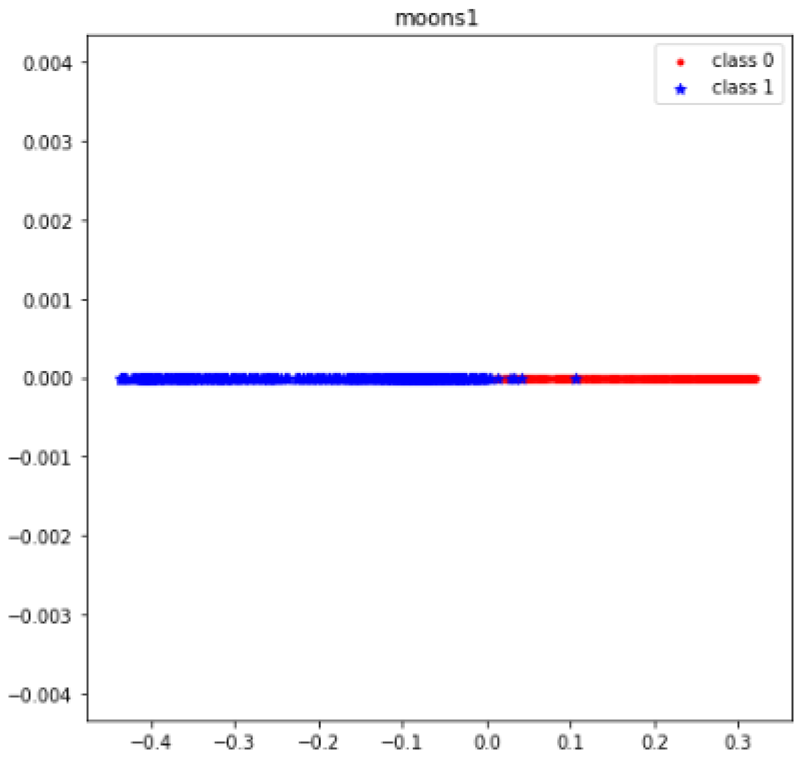
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans 1D



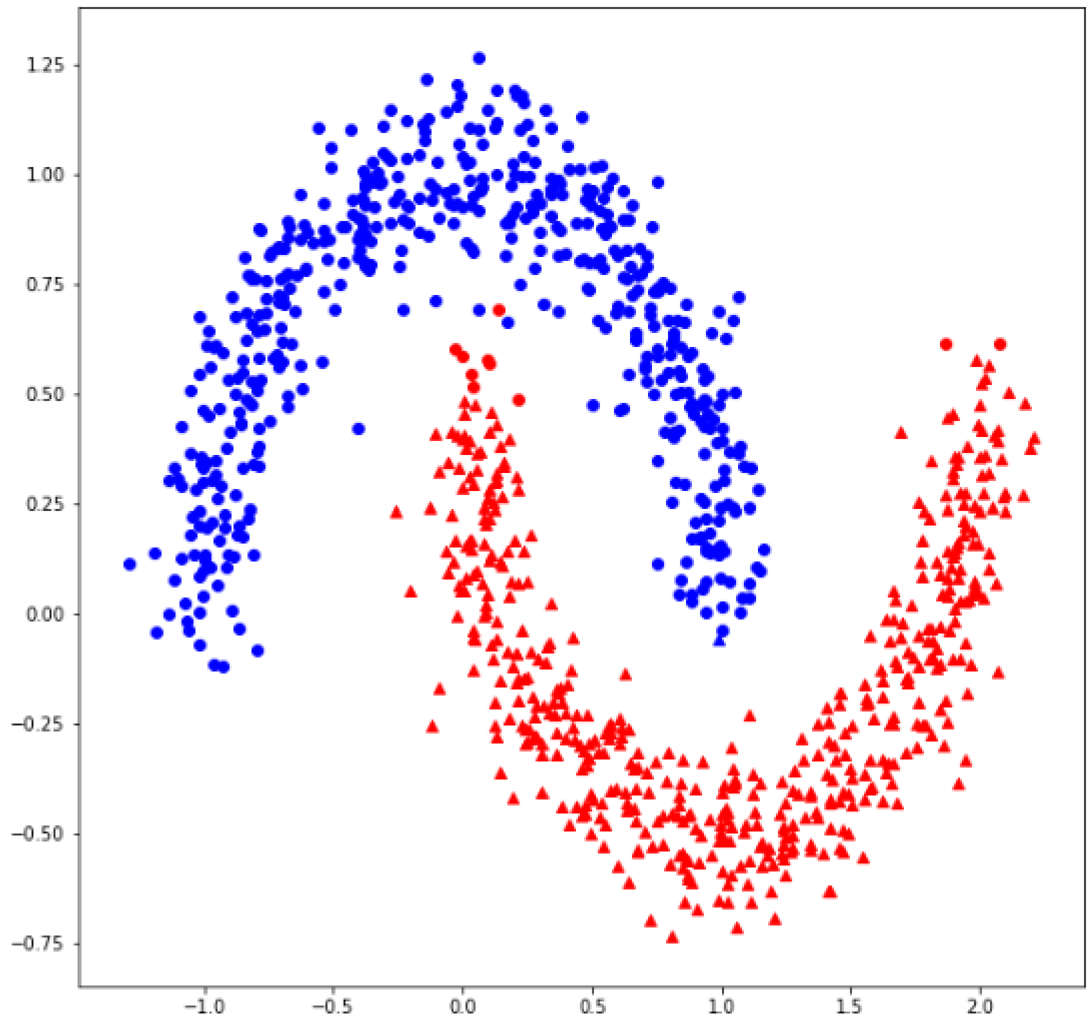
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KPCA on 1D



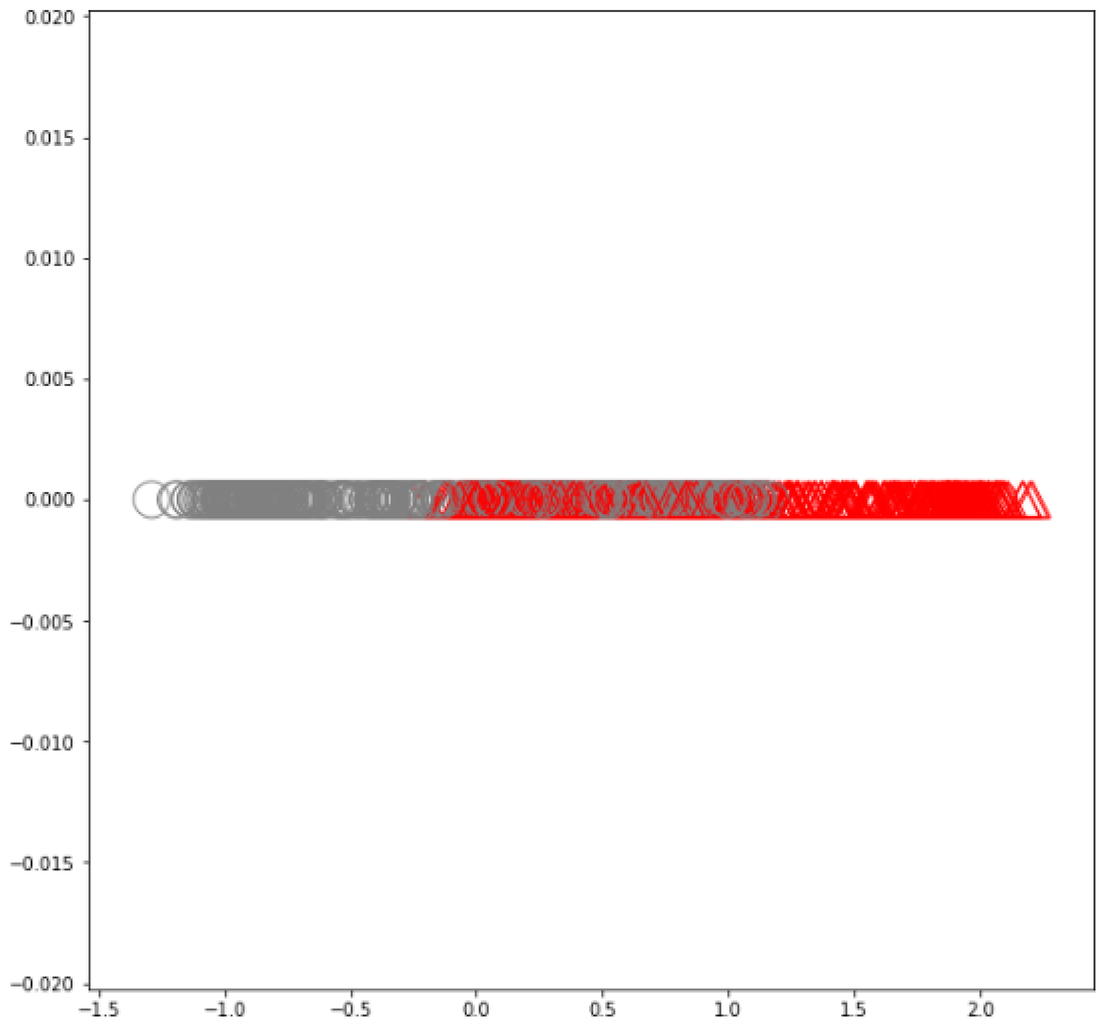
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans on KPCA



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

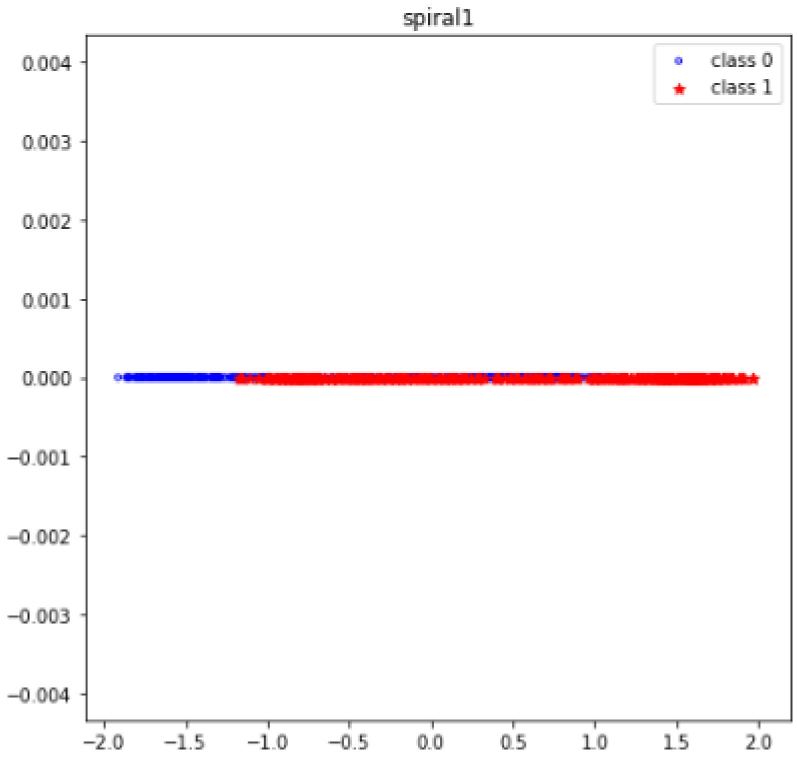
K means in 1-D



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

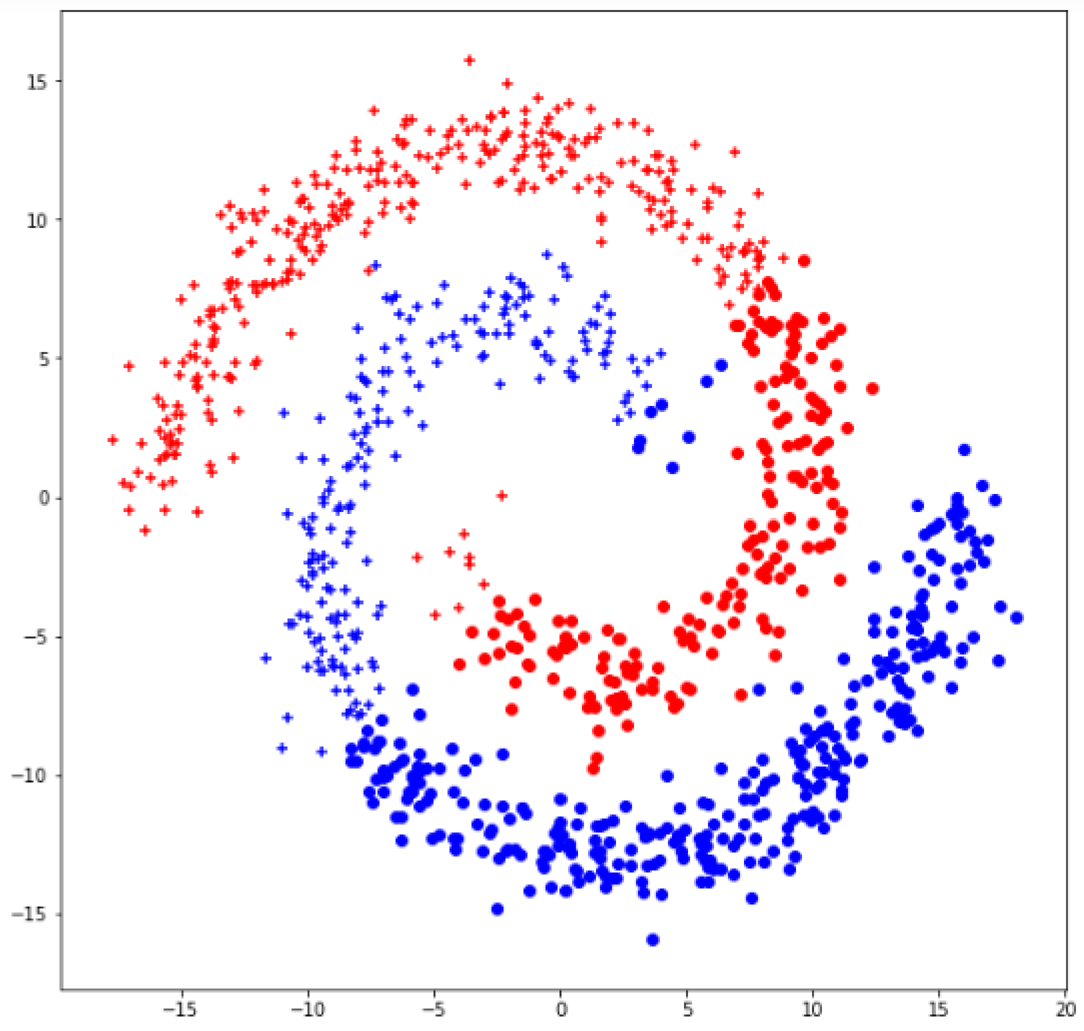
Dataset: - Spiral 1

Applying PCA on 1D



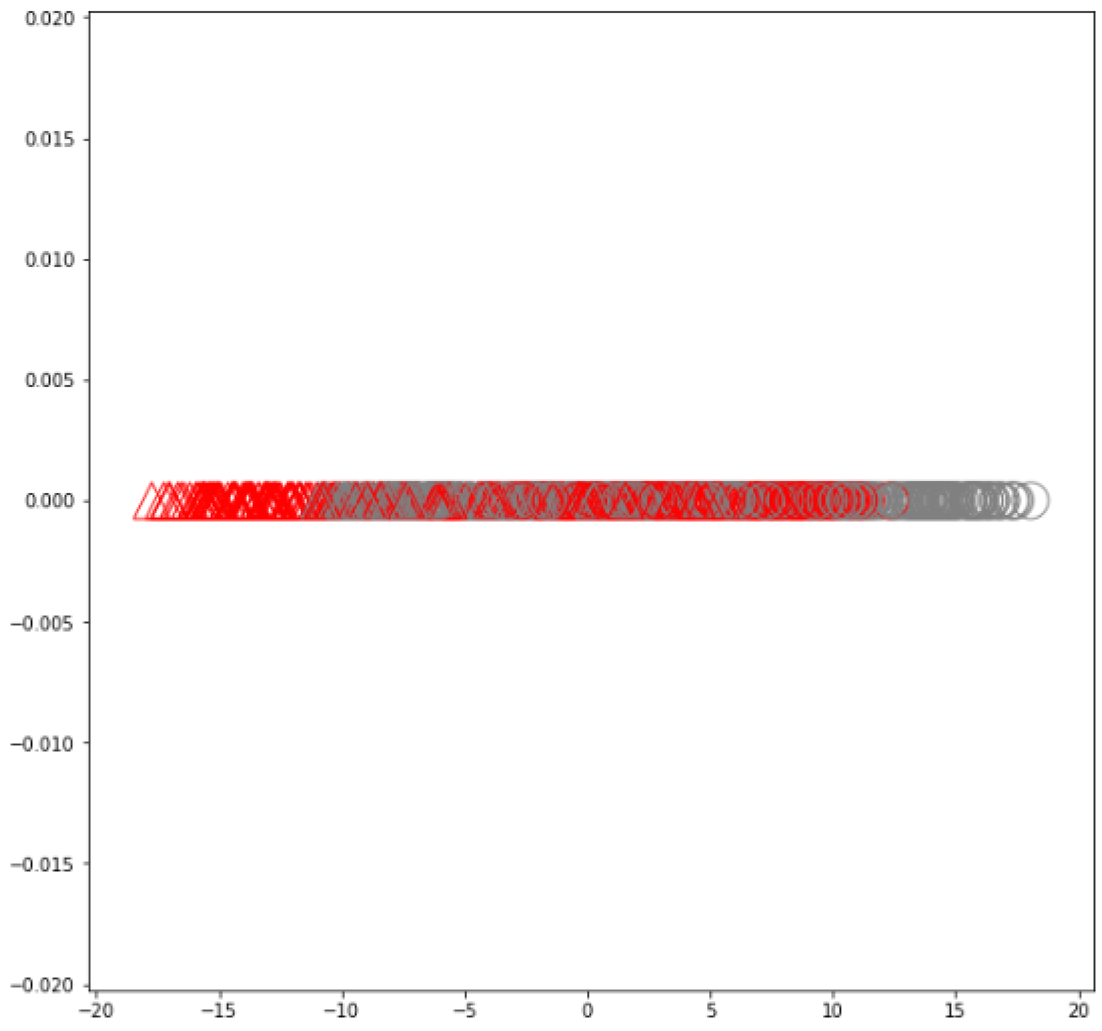
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying K Means on PCA

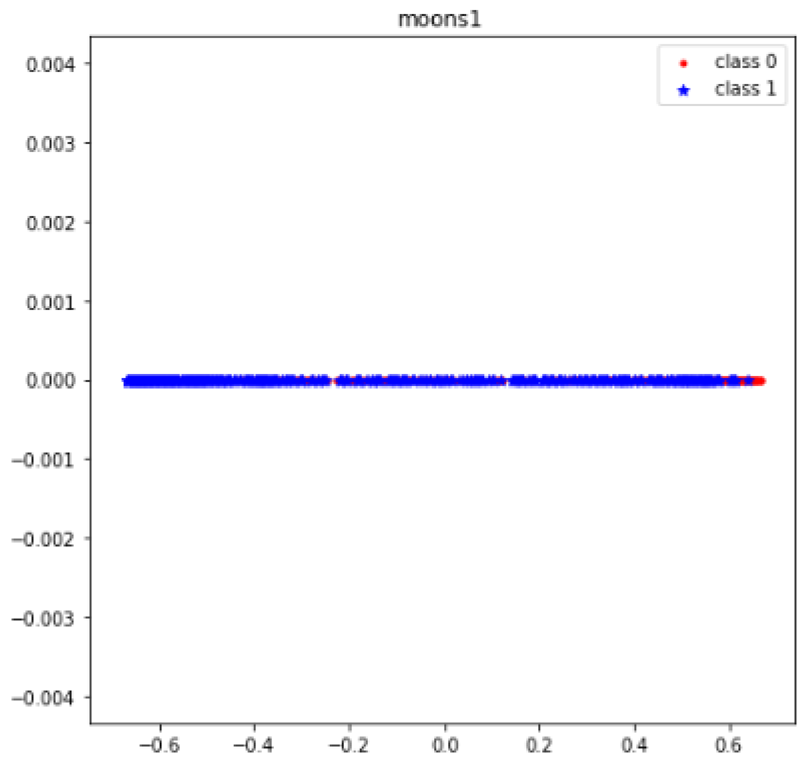


Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans 1D

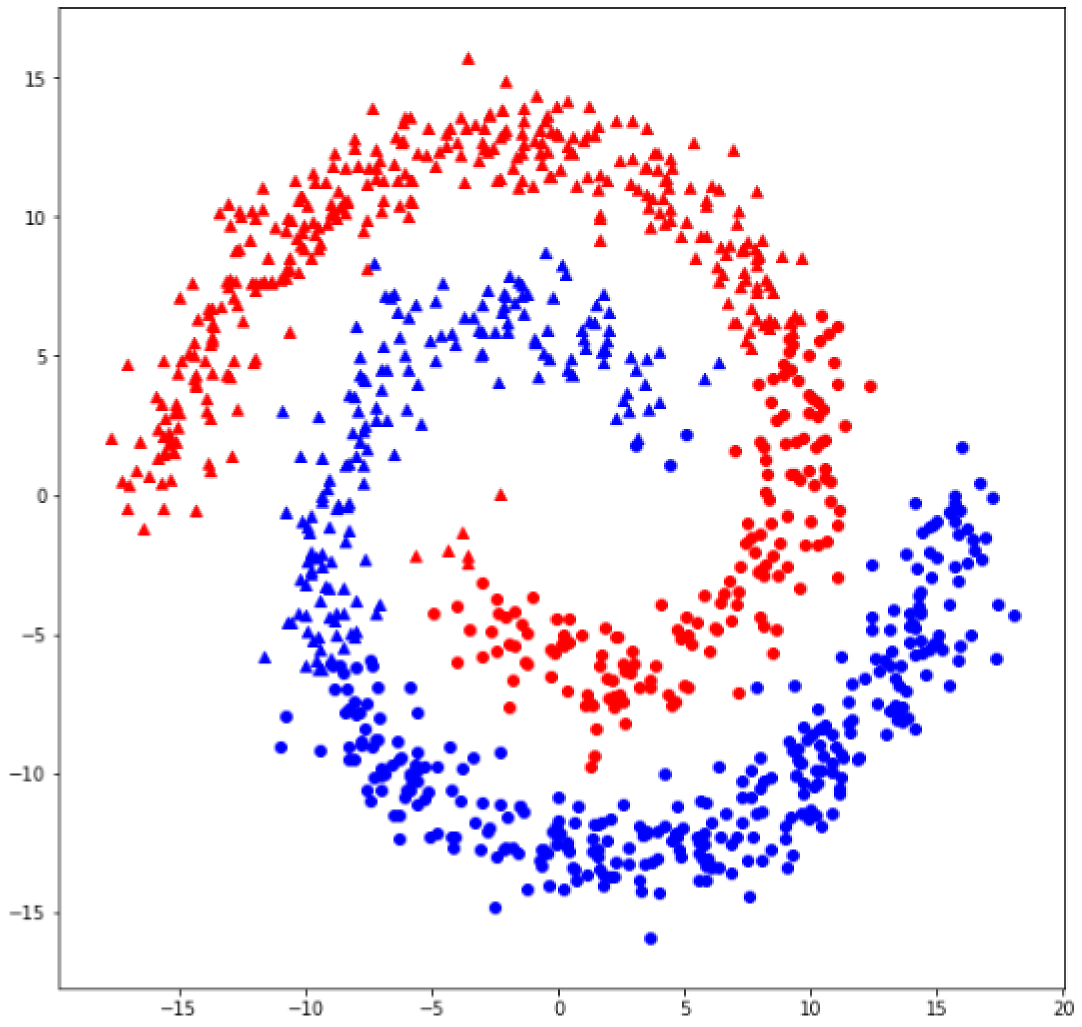


Applying KPCA on 1D



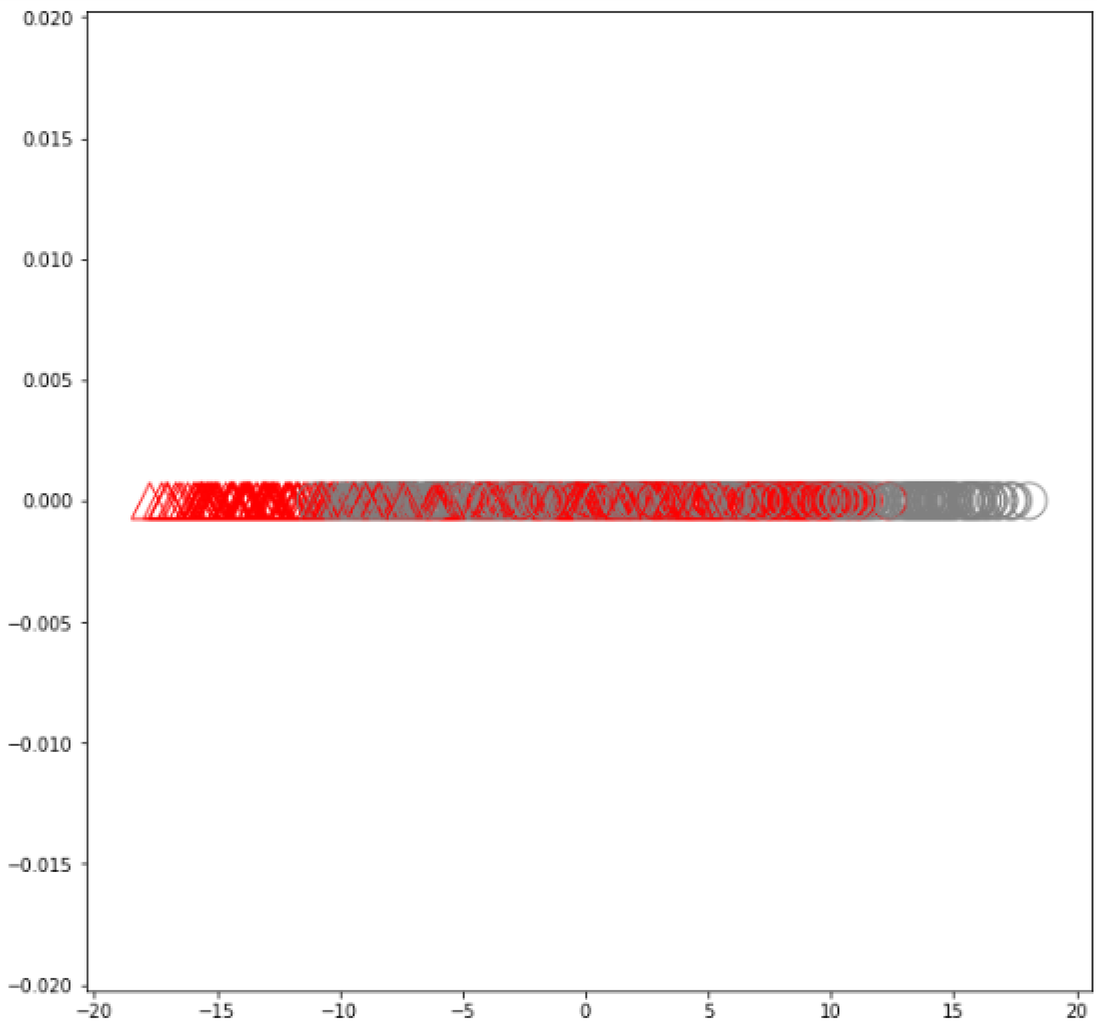
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans on KPCA



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

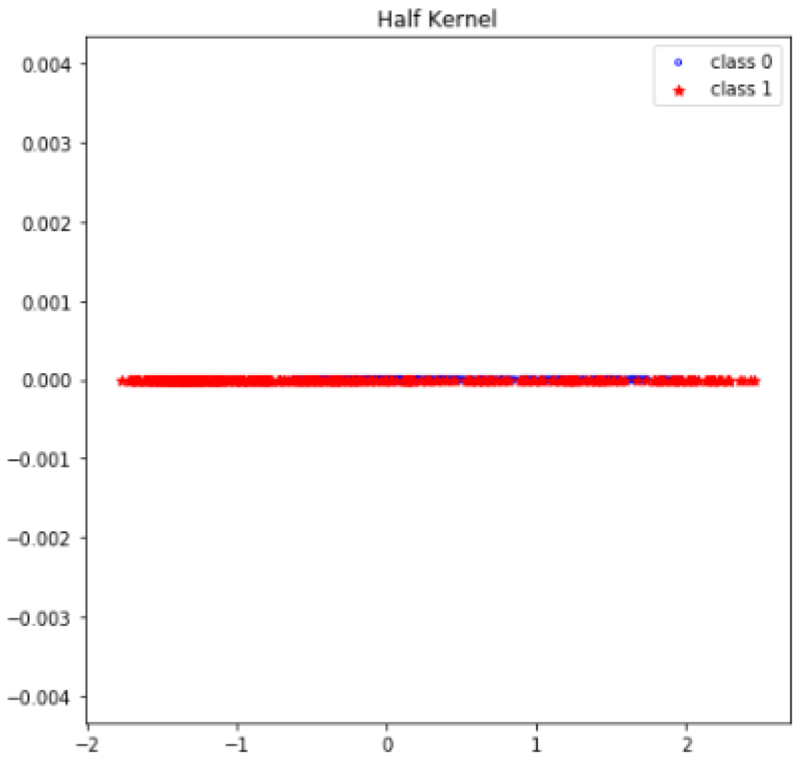
K means in 1-D



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

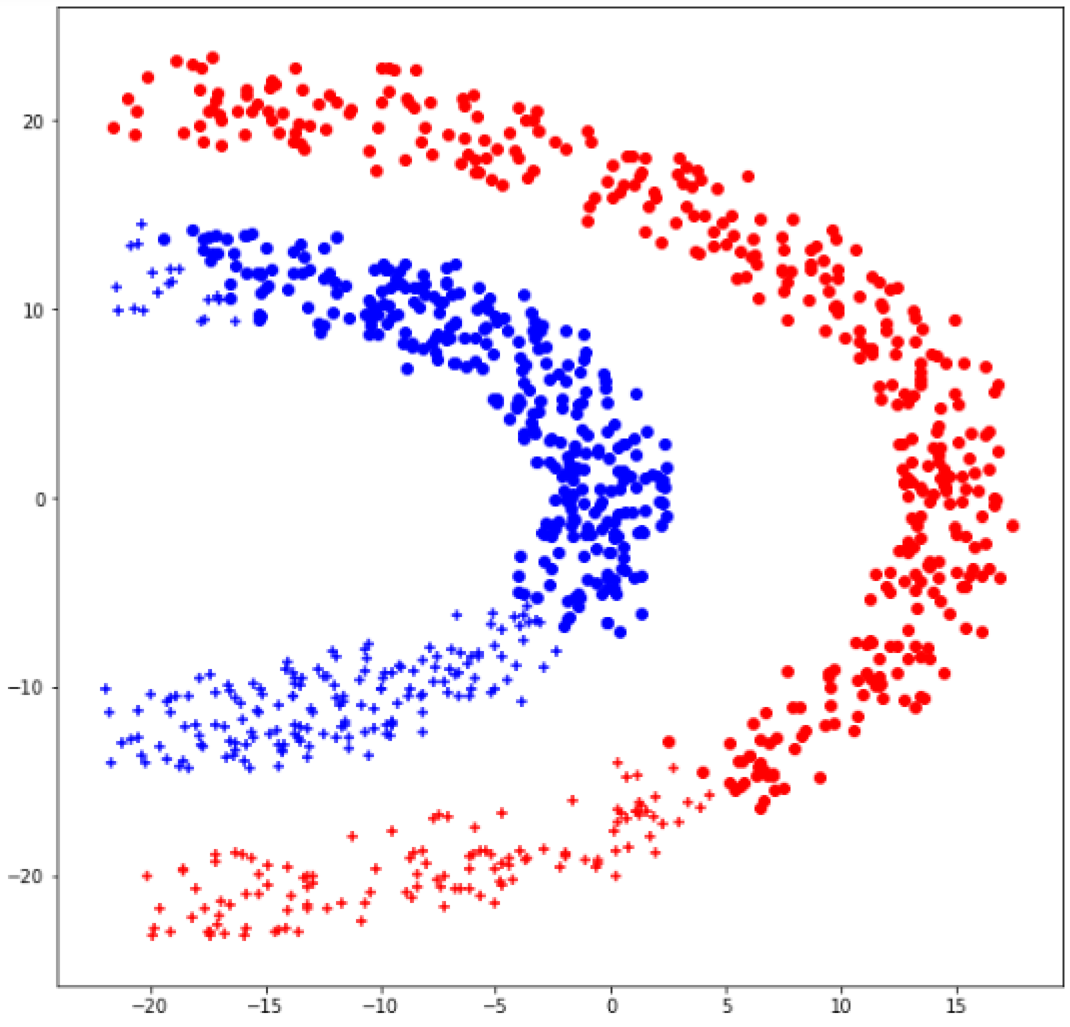
Dataset: - Half Kernel

Applying PCA on 1D



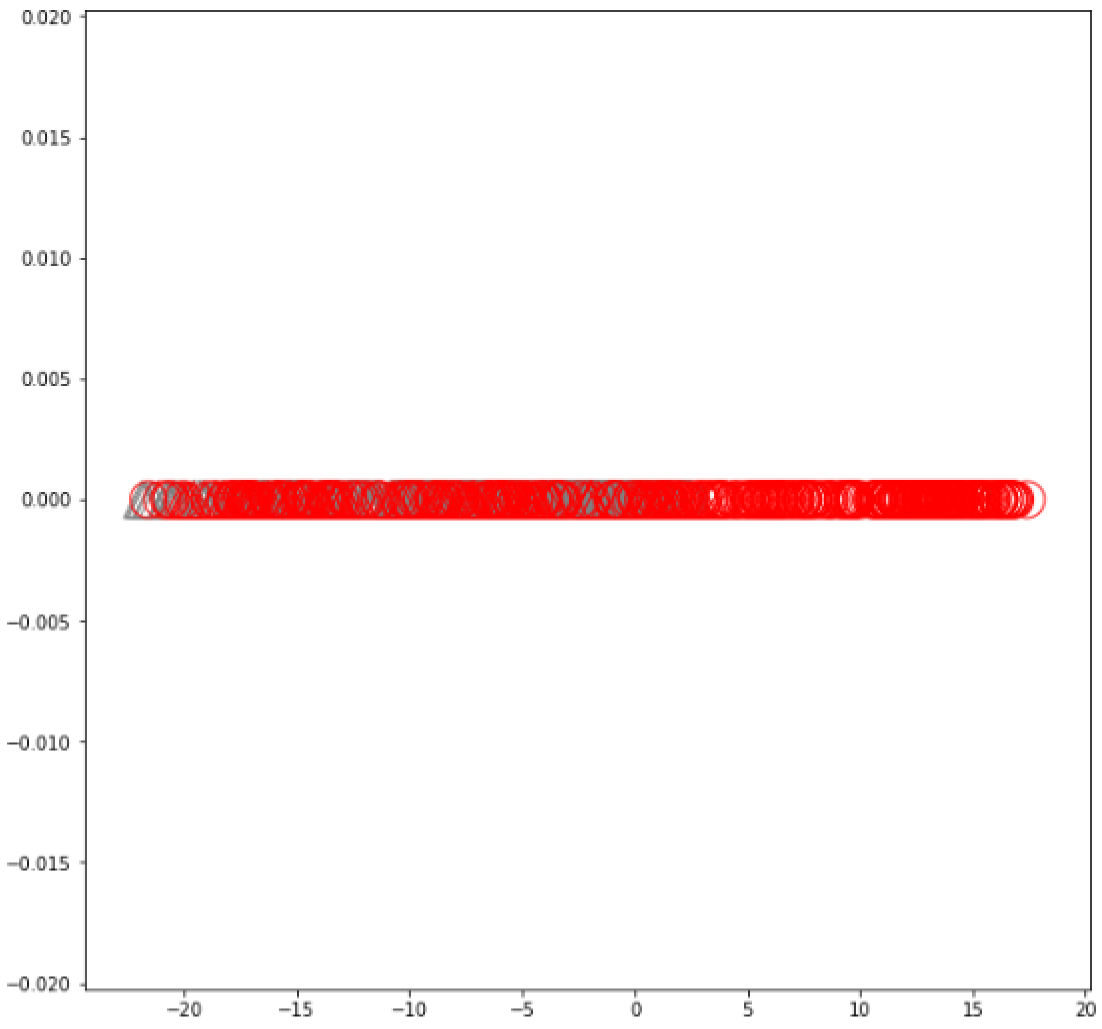
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying K Means on PCA



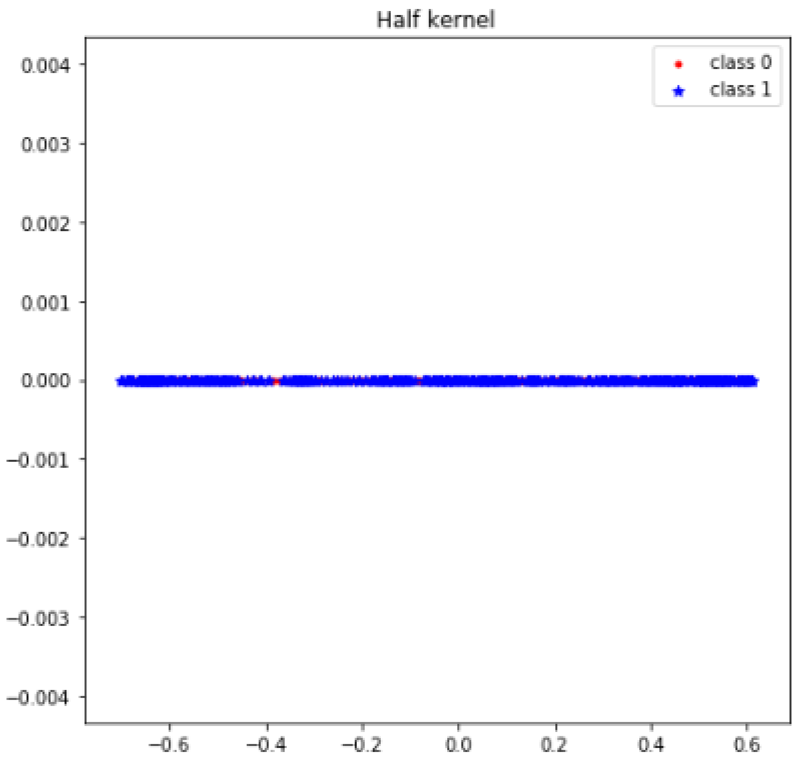
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans 1D



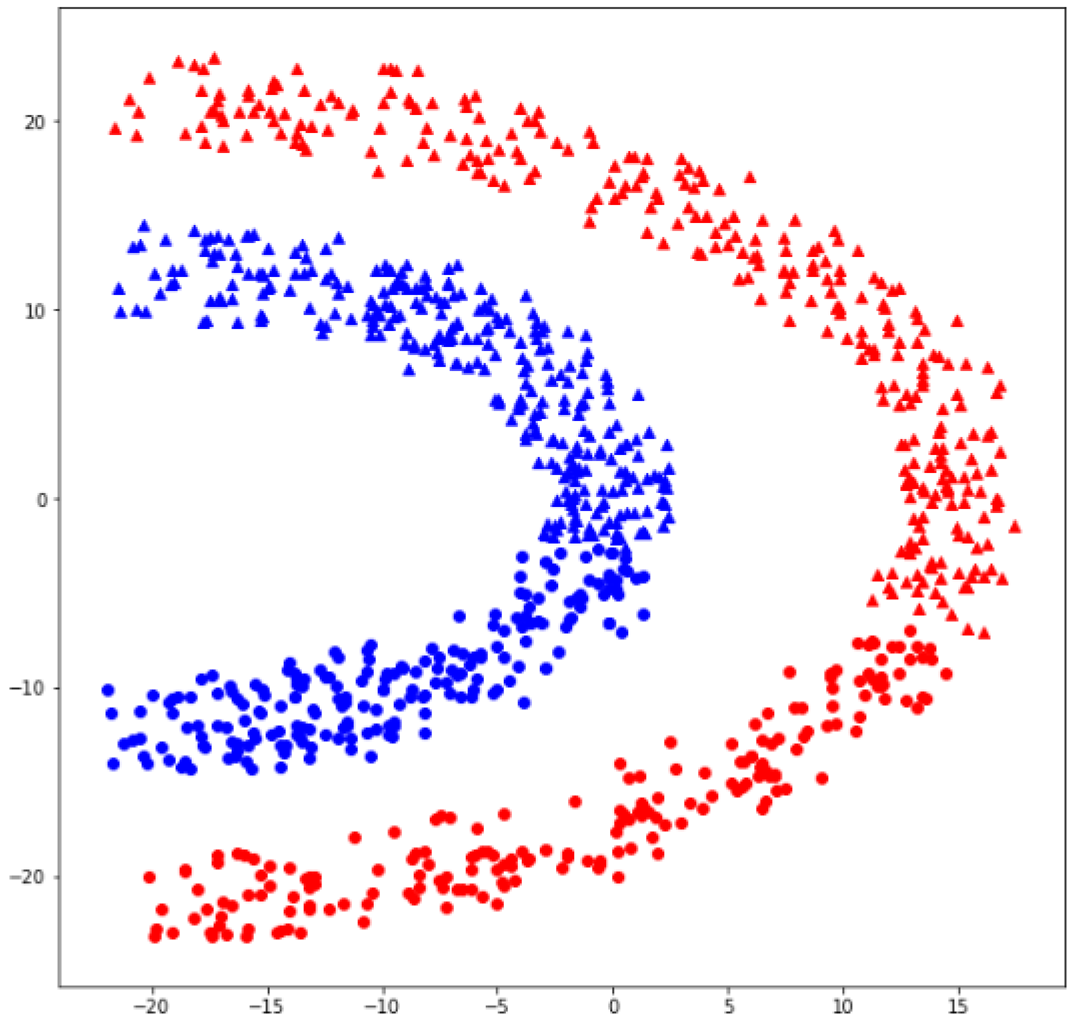
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KPCA on 1D



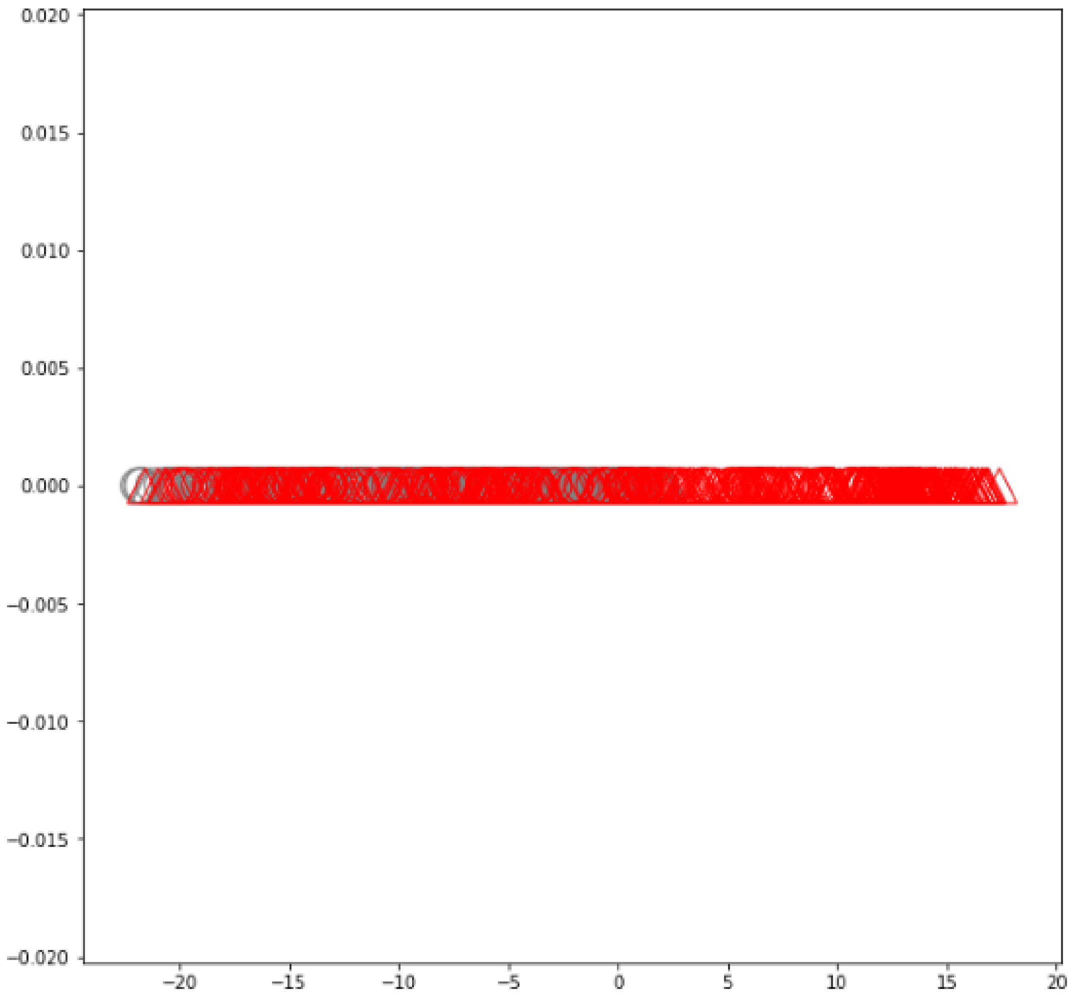
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Applying KMeans on KPCA



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

K means in 1-D



Q4) Briefly discuss the results achieved by dimensionality reduction and clustering.
A4)

PCA (Principal Component Analysis) transforms the dataset from the original coordinate system to new coordinate system where new coordinates are chosen by data itself

The general method to obtain PCA is given as follows: -

1) Step1: - Compute the mean
2) Step2: - Compute the covariance matrix
3) Step3: - Find the eigen values from largest to the smallest
4) Step4: - We determine the top 'n' eigen values
5) Step5: -Transform the data into the new space created by "n" eigen values which becomes the eigen vector

Dataset: - Circles0.3

- PCA gives poor result for circles0.3 dataset as there is loss of information from 2D to 1D
- When K means is applied to circles0.3 it gives better result compared to others
- When KPCA is applied to circles0.3 dataset gives better result than PCA because it uses kernel trick and RBF kernel (depending upon the dataset we use parameter tuning by gamma: - not in circles0.3)
- Now we apply K means on KPCA which gives the best result compared to other datasets due inherent sparsity of points

Dataset: - Moons1

- PCA gives poor result for moons1 dataset as there is loss of information from 2D to 1D
- When K means is applied to moons1 it gives better result compared to others
- When KPCA is applied to moons1 dataset gives better result than PCA because it uses kernel trick and RBF kernel (depending upon the dataset we use parameter tuning by gamma: - 18 in moons1)
- Now we apply K means on KPCA which gives the best result compared to other datasets due inherent sparsity of points

Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Dataset: - Spiral1

- PCA gives poor result for spiral1 dataset as there is loss of information from 2D to 1D
- When K means is applied to spiral1 it gives better result compared to others
- When KPCA is applied to spiral1 dataset gives moderate result than PCA because it uses kernel trick and RBF kernel (depending upon the dataset we use parameter tuning by gamma: - no gamma in spiral dataset)
- Now we apply K means on KPCA which gives the moderate result compared to other datasets due inherent sparsity of points

Dataset: - Half Kernel

- PCA gives poor result for half kernel dataset as there is loss of information from 2D to 1D
- When K means is applied to half kernel it gives less moderate result compared to others
- When KPCA is applied to spiral1 dataset gives less moderate result than PCA because it uses kernel trick and RBF kernel (depending upon the dataset we use parameter tuning by gamma: - no gamma in spiral dataset)
- Now we apply K means on KPCA which gives the less moderate result compared to other datasets due inherent sparsity of points

Summary

We first apply PCA on the given datasets and hence PCA produces dimensionality reduction from 2D to 1D which thereby produces loss of information and thus we get poor results for considerable datasets , then we apply K means on the 1D data which is a clustering technique , we do clustering to separate the data or make clusters thus a good cluster means equal separation of data in different colors and correct markers for class. In KPCA we get better result as compared to PCA because it uses kernel trick and uses curve on the graph for the data to form clusters and hence produces better result similarly k means is applied and based on the observation I can thereby summarize 4 datasets as “good”, “moderate” and “less moderate” clustering.

Comparison chart for clustering

Parameters	Circles0.3	Moons1	Spiral 1	Half Kernel
	Good	Good	Moderate	Less Moderate

The following comparison of good, moderate and less moderate is based on the type of clustering with relative comparison of each other in 2D and 1D representation.

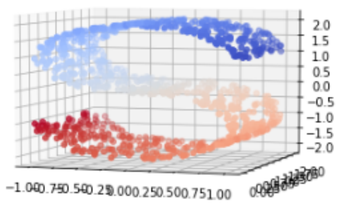
Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

Q5) Generate two types of datasets: 3D S-shape and Swiss Roll. You can use the following code for this purpose, or the one of your preference.

A5)

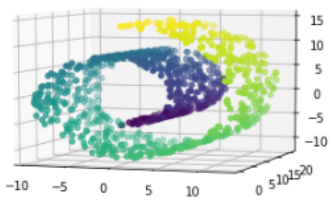
3D S Shape

```
a,color=sklearn.datasets.make_s_curve(n_samples=1000, noise=0.0, random_state=0)
fig = plt.figure()
sroll = fig.add_subplot(111, projection='3d')
sroll.scatter(a[:, 0], a[:, 1], a[:, 2], c=color, cmap=plt.cm.coolwarm)
sroll.view_init(6,-72)
```



3D Swiss Roll

```
b,color=sklearn.datasets.make_swiss_roll(n_samples=1000, noise=0.0, random_state=None)
fig = plt.figure()
swissroll = fig.add_subplot(111, projection='3d')
swissroll.scatter(b[:, 0], b[:, 1], b[:, 2], c=color, cmap=plt.cm.viridis)
swissroll.view_init(6,-72)
```

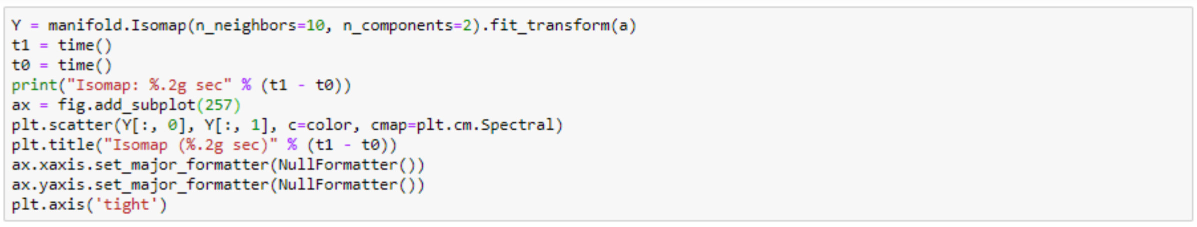


**Q6) Apply manifold learning methods: Isomap, Laplacian eigenmaps and LLE, obtaining new
"unrolled" data on the 2D space. Choose the parameters of the models (e.g., distance,
kernel, and number of neighbours) on your own and explain why you chose those. Show
the plots of both datasets before and after the embedding. Discuss the results.**

A6)

Dataset: - S Shape

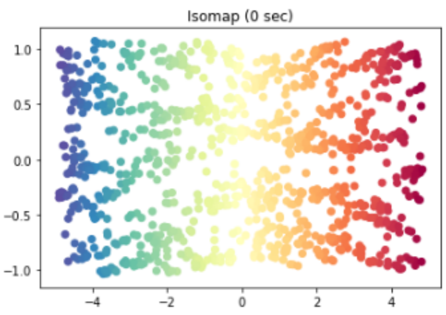
- **Isomap**



```
Y = manifold.Isomap(n_neighbors=10, n_components=2).fit_transform(a)
t1 = time()
t0 = time()
print("Isomap: %.2g sec" % (t1 - t0))
ax = fig.add_subplot(257)
plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
plt.title("Isomap (%.2g sec)" % (t1 - t0))
ax.xaxis.set_major_formatter(NullFormatter())
ax.yaxis.set_major_formatter(NullFormatter())
plt.axis('tight')
```

```
Isomap: 0 sec

(-5.388451966945234,
 5.323052265066302,
 -1.1553179205993098,
 1.1904312957413974)
```



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

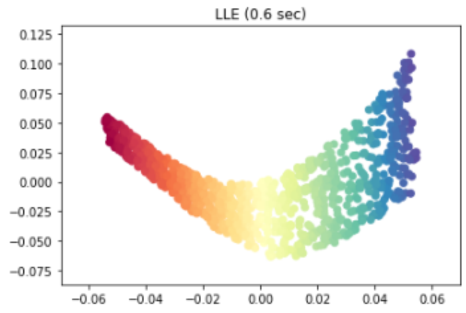
- **LLE**

```
methods = ['standard']
labels = ['LLE']
for i, method in enumerate(methods):
    t0 = time()
    Y = manifold.LocallyLinearEmbedding(n_neighbors=10, n_components=2,
                                        eigen_solver='auto',
                                        method=method).fit_transform(a)

    t1 = time()
    print("%s: %.2g sec" % (methods[i], t1 - t0))

    ax = fig.add_subplot(252 + i)
    plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
    plt.title("%s (%.2g sec)" % (labels[i], t1 - t0))
    ax.xaxis.set_major_formatter(NullFormatter())
    ax.yaxis.set_major_formatter(NullFormatter())
    plt.axis('tight')
```

```
standard: 0.6 sec
```

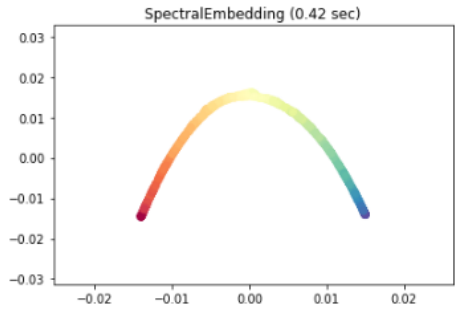


- **Laplacian Eigen Maps**

```
t0 = time()
se = manifold.SpectralEmbedding(n_components=2,
                                n_neighbors=10)
Y = se.fit_transform(a)
t1 = time()
print("SpectralEmbedding: %.2g sec" % (t1 - t0))
ax = fig.add_subplot(259)
plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
plt.title("SpectralEmbedding (%.2g sec)" % (t1 - t0))
ax.xaxis.set_major_formatter(NullFormatter())
ax.yaxis.set_major_formatter(NullFormatter())
plt.axis('tight')
```

```
SpectralEmbedding: 0.42 sec
```

```
(-0.025351240571766152,
 0.026279589868089336,
 -0.031270508654073464,
 0.03277926413797469)
```



Name: - Sami Ali Choudhry Std no: - 105187035 Course: - Pattern Recognition (Assignment 3)

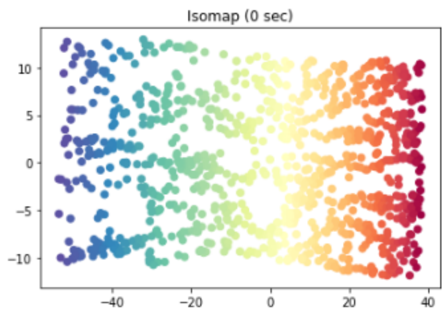
Dataset: - Swiss Roll

- ## Isomap

```
Y = manifold.Isomap(n_neighbors=10, n_components=2).fit_transform(b)
t1 = time()
t0 = time()
print("Isomap: %.2g sec" % (t1 - t0))
ax = fig.add_subplot(257)
plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
plt.title("Isomap (%.2g sec)" % (t1 - t0))
ax.xaxis.set_major_formatter(NullFormatter())
ax.yaxis.set_major_formatter(NullFormatter())
plt.axis('tight')
```

```
Isomap: 0 sec

(-58.06190571782853,
 42.96955855692277,
 -13.153241798477833,
 14.223702950028075)
```

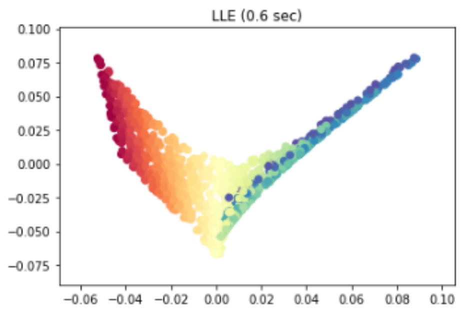


- ## LLE

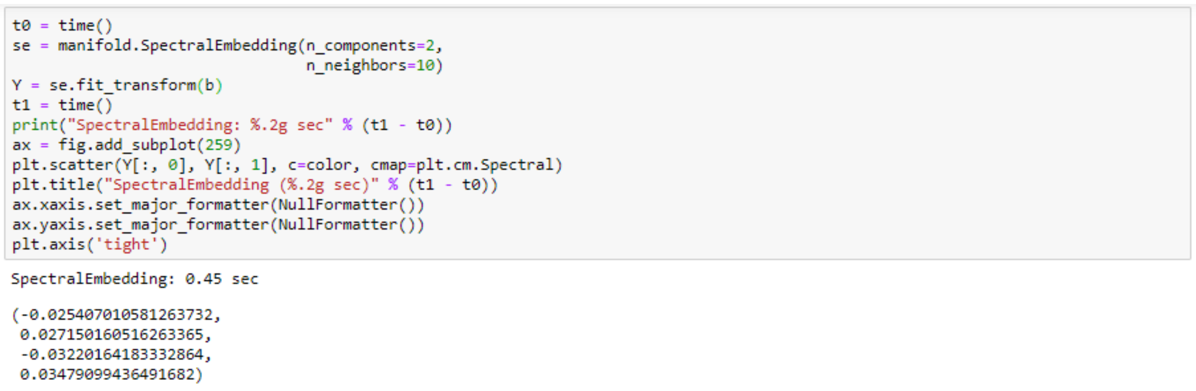
```
methods = ['standard']
labels = ['LLE']
for i, method in enumerate(methods):
    t0 = time()
    Y = manifold.LocallyLinearEmbedding(n_neighbors=10, n_components=2,
                                        eigen_solver='auto',
                                        method=method).fit_transform(b)
    t1 = time()
    print("%s: %.2g sec" % (methods[i], t1 - t0))

    ax = fig.add_subplot(252 + i)
    plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
    plt.title("%s (%.2g sec)" % (labels[i], t1 - t0))
    ax.xaxis.set_major_formatter(NullFormatter())
    ax.yaxis.set_major_formatter(NullFormatter())
    plt.axis('tight')
```

```
standard: 0.6 sec
```



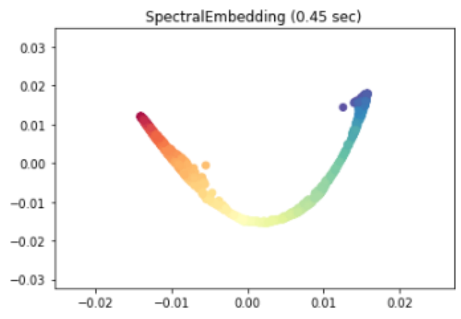
- ## Laplacian Eigen Maps



```
t0 = time()
se = manifold.SpectralEmbedding(n_components=2,
                                n_neighbors=10)
Y = se.fit_transform(b)
t1 = time()
print("SpectralEmbedding: %.2g sec" % (t1 - t0))
ax = fig.add_subplot(259)
plt.scatter(Y[:, 0], Y[:, 1], c=color, cmap=plt.cm.Spectral)
plt.title("SpectralEmbedding (%.2g sec)" % (t1 - t0))
ax.xaxis.set_major_formatter(NullFormatter())
ax.yaxis.set_major_formatter(NullFormatter())
plt.axis('tight')
```

```
SpectralEmbedding: 0.45 sec

(-0.025407010581263732,
 0.027150160516263365,
 -0.03220164183332864,
 0.03479099436491682)
```



Affinity:- We are using nearest neighbour method, thus **RBF** will provide better results for it , it does not make use of Euclidian distance between the points and hence uses Gaussian Similarity which provides better result in this case which takes the points to higher dimension and projects them back into lower dimension and gives good results

Mathematically gaussian similarity can be explained as following

D (x1, x2) = exp $(-||x1-x2||^2/\text{sigma}^2)$

N_Components: - For the coding on 2D plane we make use of **"2"** number of components

Number of neighbors: - Based on ease of computation I have selected **"10"** number of neighbours because it will help in easy calculation and also simple computation for adjacency matrix because on increasing the number of neighbors it will increase the values of adjacency matrix which will be difficult to compute and hence **time** taken by it will also increase and thereby reduce the **accuracy** of our 2d projection, thus considering the considerable and moderate value by brute force I select number of neighbors as 10.

Q7) Explain mathematically (or textually, no formal proof required) how Laplacian eigenmaps unroll the 3D S and the Swiss Roll. You may want to use some of the formulas discussed in class, if needed, but not necessarily.

A7)

Laplacian eigenmaps unrolling 3DS and Swiss Roll: -

It is a type of projection of data in lower linear dimensional data where distances are present in the original dimension. It stores the information which was present originally of the neighbourhood.

Step1: - Input X, for evert point "x" we determine the nearest neighbours

Step2: - Construct an undirected graph G = (V, E, W) where the symbols represent the following

- V = number of vertices
- E = number of edges
- W= Weight matrix using the RBF which follows gaussian similarity

Mathematically gaussian similarity can be explained as following

D (x1, x2) = exp (-||x1-x2||^2 /sigma^2)

Where weight matrix will use the two points x1 and x2 to determine the distances between them

Step3: - We determine the degree matrix where i,j will be iterated for total edges on the particular node "A"

Step4: - Using graph Laplacian we project or represent the data into the new space thus considering the neighbour references which were present in the original form are also present in the new space form thus neighbors before and after between the transformation of the points remain the same and hence using the weight matrix W, new space B is constructed which is given by B=A-W**.** We try to reduce the distance between the points in the new space in order to obtain better results

Step5: - Use the eigen values of B and project the points and arrange them from smallest to largest, thus a lower dimension projection of points is obtained and hence distances remain the same with reference to the neighbors, hence information is preserved in terms of neighbors of the points