```

import numpy as np
import matplotlib.pyplot as plt

from sklearn.decomposition import PCA, KernelPCA
from sklearn.datasets import make_circles
from mpl_toolkits.mplot3d import Axes3D
from sklearn import decomposition
import time

np.random.seed(0)

X, y = make_circles(n_samples=400, factor=.3, noise=.05)

plt.figure(1)

plt.title("Original space")
reds = y == 0
blues = y == 1


plt.scatter(X[reds, 0], X[reds, 1], c="red",
            s=20, edgecolor='k')
plt.scatter(X[blues, 0], X[blues, 1], c="blue",
            s=20, edgecolor='k')




#PCA 2d -> 1d original points

points_orig=np.zeros((X.shape[0],2))

points_orig[:,0]=X[:,0]
points_orig[:,1]=X[:,1]

print("Calculating PCA 2d -> 1d, orig. points ", end="", flush=True)
start_time = time.time()
pca1_orig = decomposition.PCA(n_components=1)
pca1_orig.fit(points_orig)
points_pca1_orig_reduced = pca1_orig.transform(points_orig)
print("\t\t took %s seconds " % round((time.time() - start_time),5))



plt.figure(14)
plt.title("Original 2d points -> 1d")
plt.scatter(points_pca1_orig_reduced[reds, 0], np.zeros((points_pca1_orig_reduced[reds, 0]).shape[0]),
c="red",
            s=20, edgecolor='k')


plt.scatter(points_pca1_orig_reduced[blues, 0], np.zeros((points_pca1_orig_reduced[blues, 0]).shape[0]),
c="blue",
            s=20, edgecolor='k')




fig2 = plt.figure(2)

ax2 = fig2.add_subplot(111, projection='3d')
ax2.set_title("2d points -> 3d points")


# \phi(x) = (x1^2, x_2^2, \sqrt(2) x1x2)
#Z=np.sqrt(2)*X[:,0]*X[:,1]
#X[:,0]=X[:,0]**2;
#X[:,1]=X[:,1]**2;

# \phi(x) = (x1, x_2,  exp((x1^2+x2^2)/1.25))
Z=np.exp( ((X[:,0]**2)+X[:,1]**2)/1.25)



ax2.scatter(X[reds,0],X[reds,1],Z[reds],s=20, color='red')
ax2.scatter(X[blues,0],X[blues,1],Z[blues],s=20, color='blue')

```

```
79
80  points=np.zeros((X.shape[0],3))
81
82  points[:,0]=X[:,0]
83  points[:,1]=X[:,1]
84  points[:,2]=Z
85
86
87  print("Calculating PCA to 2d...", end="", flush=True)
88  start_time = time.time()
89  pca2 = decomposition.PCA(n_components=2)
90  pca2.fit(points)
91  points_pca2_reduced = pca2.transform(points)
92  print("\t\t took %s seconds " % round((time.time() - start_time),5))
93
94  plt.figure(13)
95  plt.title("3d points -> PCA to 2d")
96  plt.scatter(points_pca2_reduced[reds, 0], points_pca2_reduced[reds, 1], c="red",
97              s=140, edgecolor='k')
98  plt.scatter(points_pca2_reduced[blues, 0], points_pca2_reduced[blues, 1], c="blue",
99              s=20, edgecolor='k')
100
101
102
103
104
105
106 print("Calculating PCA to 1d...", end="", flush=True)
107 start_time = time.time()
108 pca1 = decomposition.PCA(n_components=2)
109 pca1.fit(points)
110 points_pca1_reduced = pca1.transform(points)
111 print("\t\t took %s seconds " % round((time.time() - start_time),5))
112
113
114
115
116
117 plt.figure(4)
118 plt.title("3d points -> PCA to 1d")
119 plt.scatter(points_pca1_reduced[reds, 0], np.zeros((points_pca1_reduced[reds, 0]).shape[0]), c="red",
120             s=20, edgecolor='k')
121 plt.scatter(points_pca1_reduced[blues, 0], np.zeros((points_pca1_reduced[blues, 0]).shape[0]), c="blue",
122             s=20, edgecolor='k')
123
124
125
126
127
128 plt.show()
129
130
```