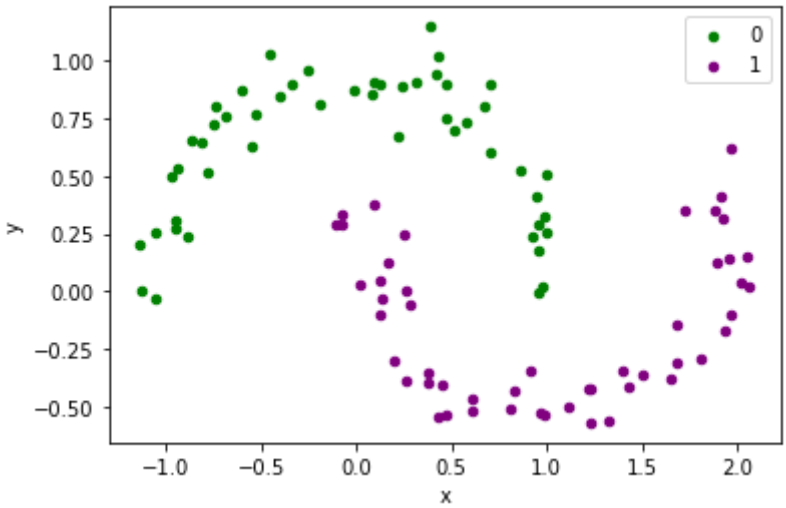
```
#generating both 2 moon and 2 circles datasets from scratch
#2 moon dataset
```

```
#importing important libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#then we import the make moons function from sklearn library to generate the dataset
from sklearn.datasets import make_moons
```

```
#now we make moons dataset with a total of 100 samples and the noise of std=0.1 as per the
X,y=make_moons(n_samples=100,noise=0.1)
```

```
#now we have the moon dataset generated,next up we can plot it for better visualization

#there we go

df=pd.DataFrame(dict(x=X[:,0],y=X[:,1],label=y))
colors={0:'green',1:'purple'}
figure,ax=plt.subplots()
grouped_df=df.groupby('label')
for i,j in grouped_df:
  j.plot(ax=ax,kind='scatter',x='x',y='y',label=i,color=colors[i])
plt.show()
```

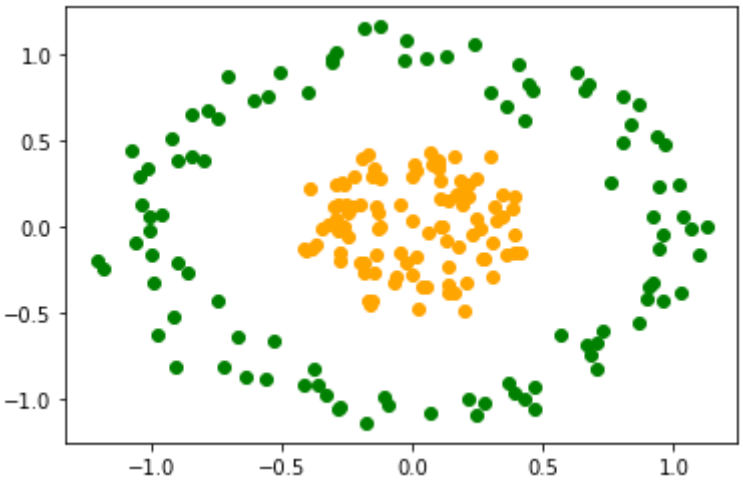



```
#using sklearn train_test_split function to split the dataset into training and testing da
from sklearn.model_selection import train_test_split
target_var=df.label
df.drop(['label'],axis=1,inplace=True)
y=target_var
x=df
```

```
###next up is generating the 2 circles dataset
#we import the following libraries
#pandas,numpy,seaborn,matplotlib as for the make moon datasets above
#since we already imported the libraries above ,the only different function we need to imp
#import is the make circles function from the sklearn library as below
from sklearn.datasets import make_circles
```

```
#lets now generate the circles dataset with 200 samples and noise as 0.1 as per the docume
X1,y1=make_circles(n_samples=200,noise=0.1,factor=.3)   #you can run a samples of higher v
```

```
# #lets plot the 2 circles dataset for better visualization
df1=pd.DataFrame(dict(x=X1[:,0],y=X1[:,1],label=y1))
c1 = X1[y1==0]
c2  = X1[y1==1]

plt.figure()
plt.scatter(c1[:, 0], c1[:, 1], color='green')
plt.scatter(c2[:, 0], c2[:, 1], color='orange')
plt.show()
```




```
#splitting the circles dataset for training and testing  using the sklearn train test spli
from sklearn.model_selection import train_test_split
target1_var=df1.label
df1.drop(['label'],axis=1,inplace=True)
y1=target1_var
x=df1
```

```
####now that we have the 2 datasets ready to go,,next thing we need to do for both dataset
#find a parameter setting on both datasets for which the kernel PCA will successfully sepa
#2 clusters or basically what we call components
#so how do we go about it?
#we proceed as below
```

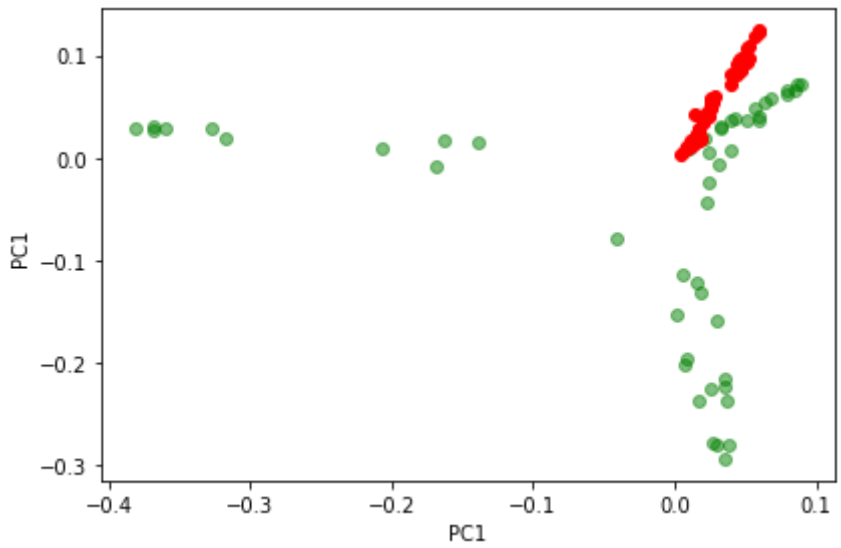
```
#our main task now is to know how implement the kernel PCA function which we gonna be usir
#and investigate where it succeeds and fails
#we can as well use the sklearn kernelPCA function to perfom the same operation (from skle
```

```
#buh in this case i gonna prefer the implementation from scratch so as make it easy inspec
# and fails
```

```
#importing some important libraries
from scipy.linalg import eigh
from scipy.spatial.distance import pdist, squareform
from scipy import exp
def kernel_pca_func(X, gamma_value, n_components):
    squared_distances = pdist(X, 'sqeuclidean')  #calculating the pairwise squared euclide
    matrix_square_distances = squareform(squared_distances)    #converting the distances i
    H = exp(-gamma_value * matrix_square_distances)
    n_val = H.shape[0]  #getting the center of the matrix
    n = np.ones((n_val,n_val)) / n_val
    H = H - n.dot(H) - H.dot(n) + n.dot(H).dot(n)
    eigenvalues, eigenvectors = eigh(H)  #centered matrix eigenpairs
    eigenvalues, eigenvectors = eigenvalues[::-1], eigenvectors[:, ::-1]
    stacked_val = np.column_stack([eigenvectors[:, i] for i in range(n_components)])
    return stacked_val
```

```
##separating the 2 moon dataset
kernel_PCA_X = kernel_pca_func(X, gamma_value=20, n_components=2) #setting the parameter g
plt.scatter(kernel_PCA_X[y==0, 0], kernel_PCA_X[y==0, 1],color='green', alpha=0.5)
plt.scatter(kernel_PCA_X[y==1, 0], kernel_PCA_X[y==1, 1],color='red' )
plt.xlabel('PC1')
plt.ylabel('PC1')
plt.tight_layout()
plt.show()
```

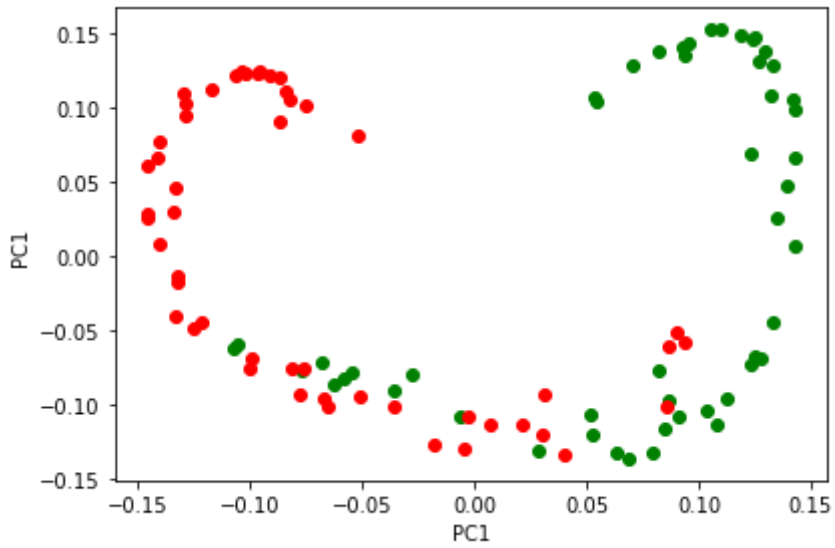
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning:
```




```
#so actually under these above  parameter setting the separation actually succeeds

#we next gonna obtain some other parameter setting for which this separation fails as belo
```

```
#we next gonna obtain some other parameter setting for which this separation fails as belo
##separating the 2 moon dataset
kernel_PCA_X = kernel_pca_func(X, gamma_value=2, n_components=2) #setting the parameter ga
plt.scatter(kernel_PCA_X[y==0, 0], kernel_PCA_X[y==0, 1],color='green')
plt.scatter(kernel_PCA_X[y==1, 0], kernel_PCA_X[y==1, 1],color='red')
plt.xlabel('PC1')
plt.ylabel('PC1')
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning:
```

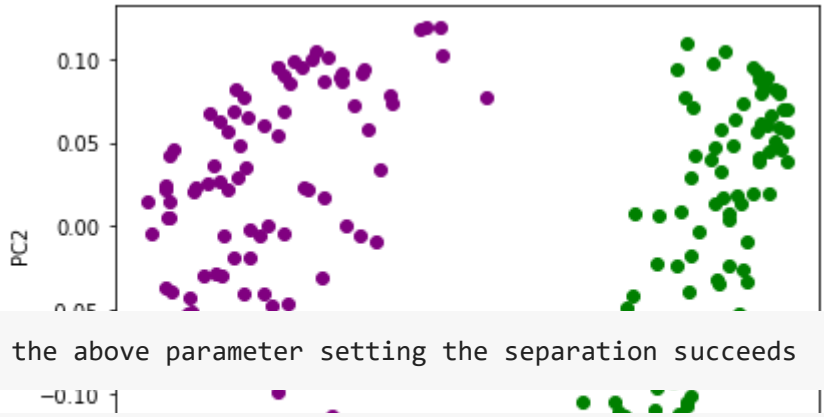



```
#so with the above parameter setting the separation fails for the 2 moon dataset
```

```
###we gonna do the same for the 2 circles dataset
```

```
##separating the 2 circles dataset
kernel_PCA_X2 = kernel_pca_func(X1, gamma_value=2, n_components=2)##we will be altering th
#actually succeeds and likewise fails
plt.scatter(kernel_PCA_X2[y1==0, 0], kernel_PCA_X2[y1==0, 1],color='green')
plt.scatter(kernel_PCA_X2[y1==1, 0], kernel_PCA_X2[y1==1, 1],color='purple')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.tight_layout()
plt.show()
```

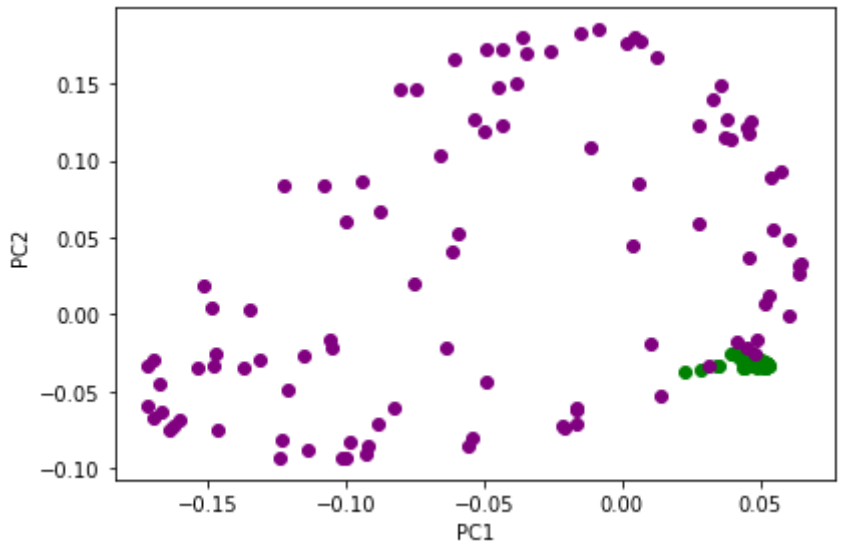
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning:
```




```
#with the above parameter setting the separation succeeds
```

```
#we see similarly a parameter setting for which the 2-circles-dataset separation fails
kernel_PCA_X2 = kernel_pca_func(X1, gamma_value=12, n_components=2)##we will be altering t
#actually succeeds and likewise fails
plt.scatter(kernel_PCA_X2[y1==0, 0], kernel_PCA_X2[y1==0, 1],color='green')
plt.scatter(kernel_PCA_X2[y1==1, 0], kernel_PCA_X2[y1==1, 1],color='purple')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning:
```




```
#reasons for failure
#the major reason why the kernlPca fails to attain good data separation is simply because
#take class labels as defined in generation of the datasets into account, meaning that if
#to compute the reconstruction error  which is used as a models measure of perfomance ,the
#not comparable between different kernels .i.e different kernels corresponds to different
```

END OF IMPLEMENTATION AND TESTING . THANK YOU!!!

✓ 0s completed at 12:45 PM