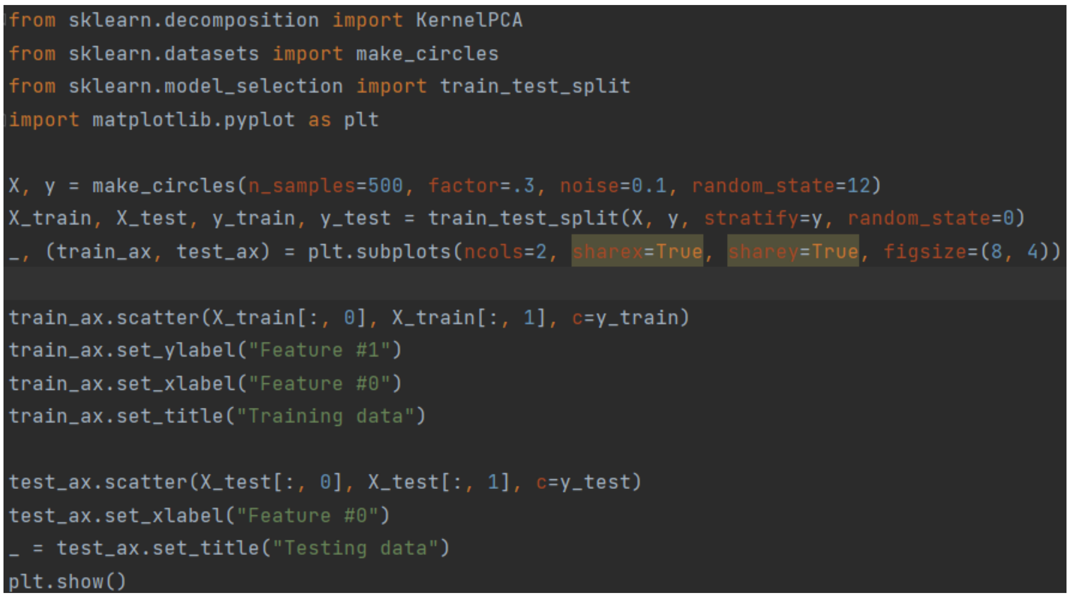
COMP 9417 – Machine Learning

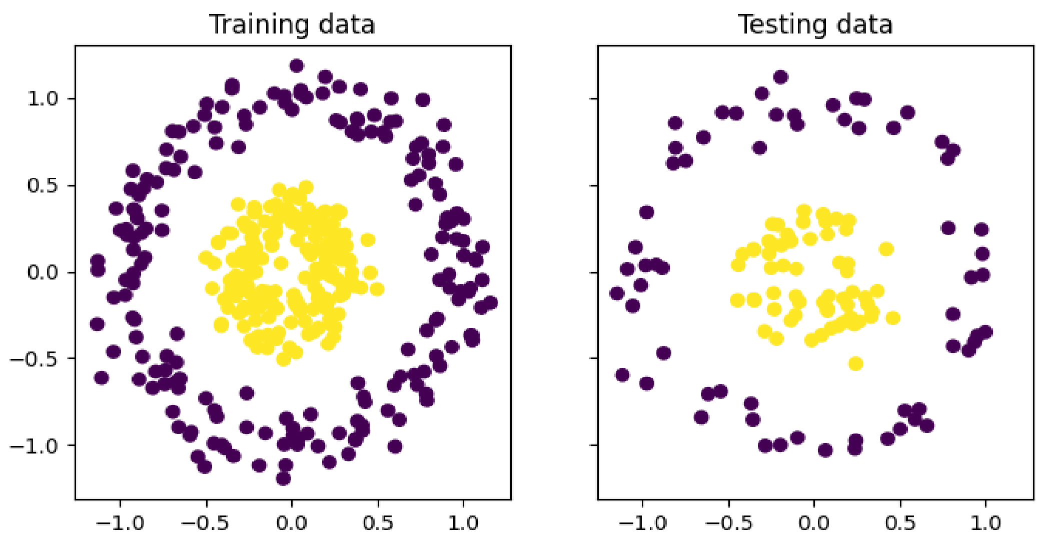
Homework 8: Unsupervised Learning

Wanqing Yang – z5325987

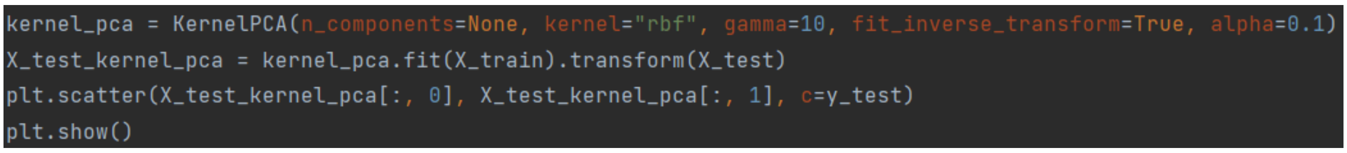
Question 1



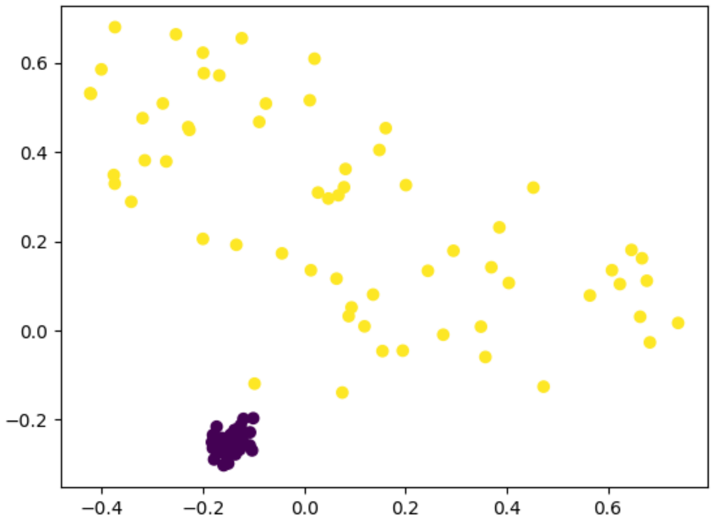
```python
from sklearn.decomposition import KernelPCA
from sklearn.datasets import make_circles
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

X, y = make_circles(n_samples=500, factor=.3, noise=0.1, random_state=12)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=0)
_, (train_ax, test_ax) = plt.subplots(ncols=2, sharex=True, sharey=True, figsize=(8, 4))

train_ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
train_ax.set_ylabel("Feature #1")
train_ax.set_xlabel("Feature #0")
train_ax.set_title("Training data")

test_ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
test_ax.set_xlabel("Feature #0")
_ = test_ax.set_title("Testing data")
plt.show()
```



Question 2



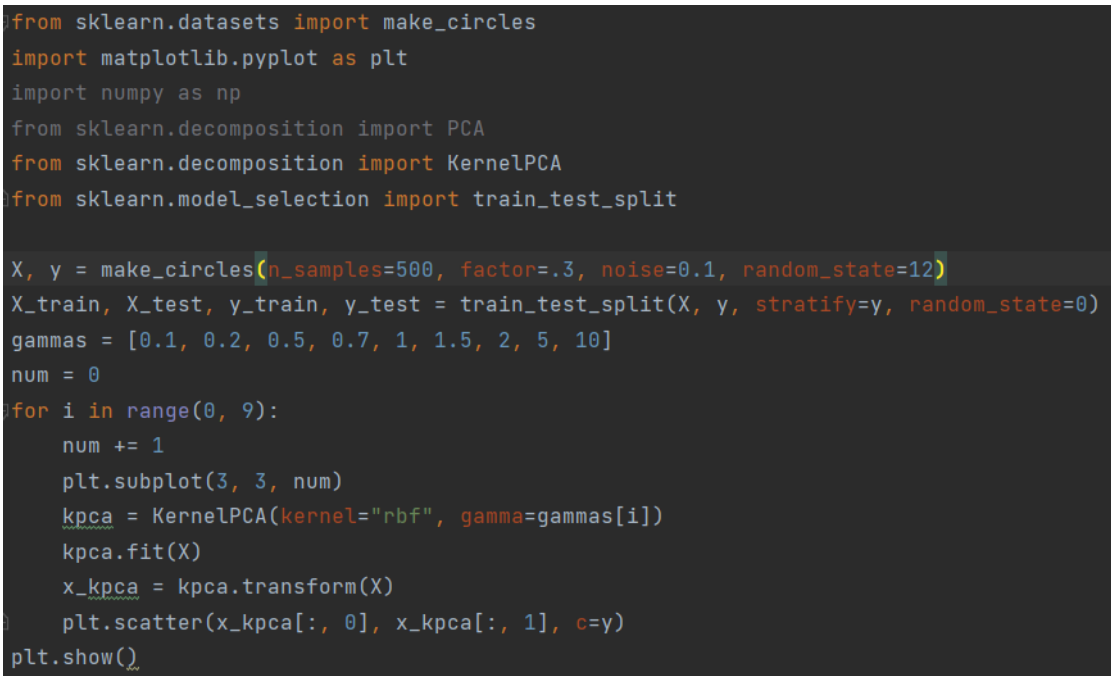
```
kernel_pca = KernelPCA(n_components=None, kernel="rbf", gamma=10, fit_inverse_transform=True, alpha=0.1)
X_test_kernel_pca = kernel_pca.fit(X_train).transform(X_test)
plt.scatter(X_test_kernel_pca[:, 0], X_test_kernel_pca[:, 1], c=y_test)
plt.show()
```



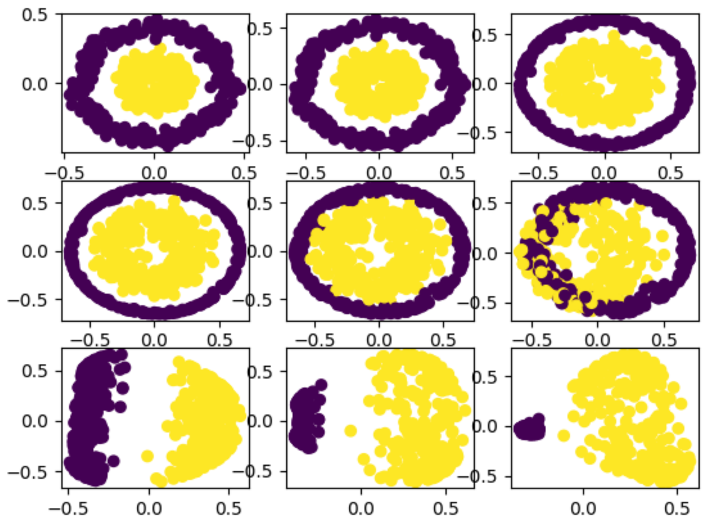
Question 3

For traditional PCA, the mapping of the same dimension only transforms the rotation of the direction, while KernelPCA, because of the additional of kernel transform, the mapping from low dimension to high dimension is generated, and then the dimension is reduced to low dimension. This process adds nonlinear transformation, and then produces different effects.

Question 4

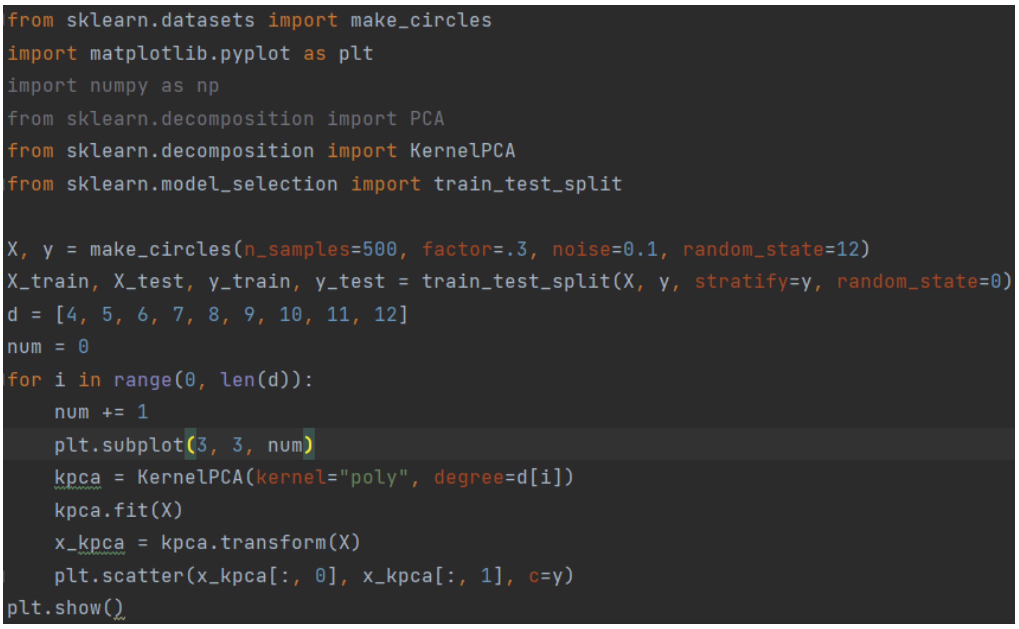


```
from sklearn.datasets import make_circles
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
from sklearn.model_selection import train_test_split

X, y = make_circles(n_samples=500, factor=.3, noise=0.1, random_state=12)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=0)
gammas = [0.1, 0.2, 0.5, 0.7, 1, 1.5, 2, 5, 10]
num = 0
for i in range(0, 9):
    num += 1
    plt.subplot(3, 3, num)
    kpca = KernelPCA(kernel="rbf", gamma=gammas[i])
    kpca.fit(X)
    x_kpca = kpca.transform(X)
    plt.scatter(x_kpca[:, 0], x_kpca[:, 1], c=y)
plt.show()
```

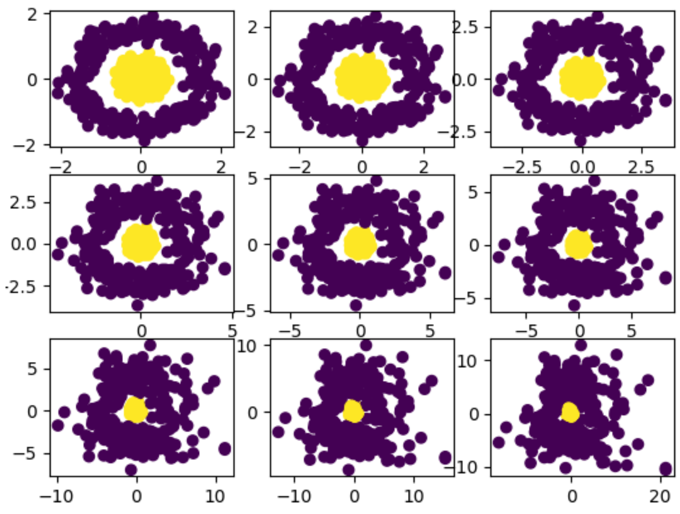


I think gamma=5 works best because it has most obvious boundaries.

Question 5



```
from sklearn.datasets import make_circles
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
from sklearn.model_selection import train_test_split

X, y = make_circles(n_samples=500, factor=.3, noise=0.1, random_state=12)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=0)
d = [4, 5, 6, 7, 8, 9, 10, 11, 12]
num = 0
for i in range(0, len(d)):
    num += 1
    plt.subplot(3, 3, num)
    kpca = KernelPCA(kernel="poly", degree=d[i])
    kpca.fit(X)
    x_kpca = kpca.transform(X)
    plt.scatter(x_kpca[:, 0], x_kpca[:, 1], c=y)
plt.show()
```



Compare with polynomial kernel, Gaussian kernel has more obvious boundary and is more suitable for PCA kernel.