



**《模式识别与机器学习》实验报告**

**学 院**：

**班 级**：

**学 号**：

**姓 名**：

**指导教师**：

教 务 处

2022 年 11 月

题目

查看如显然数据集以及相关信息，并进行数据集划分，然后使用SVC支持向量机训练，最后查看预测得分

代码

from sklearn.datasets import load\_breast\_cancer

from sklearn import svm

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

import scipy.io as scio

# 读取数据

# data = scio.loadmat("breast\_cancer.mat")

# labels = data['labels'][0]

# data1 = data['data']

cancer = load\_breast\_cancer()

X = cancer.data

y = cancer.target

print("size of the data", X.shape, y.shape)

#标签名称查看

print("target\_names:", cancer.target\_names)

#特征数量和名称查看

print('feature\_names:', cancer.feature\_names, len(cancer.feature\_names))

#查看阳性和阴性样本

print("Num of two classes:", y[y==0].shape, y[y==1].shape)

# 数据集划分

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=666,shuffle=True)

# 支持向量机的分类器的声明

model = svm.SVC(kernel='poly', degree=2)

clf = model.fit(X\_train, y\_train)

# 测试并输出结果

print(clf.score(X\_test, y\_test))

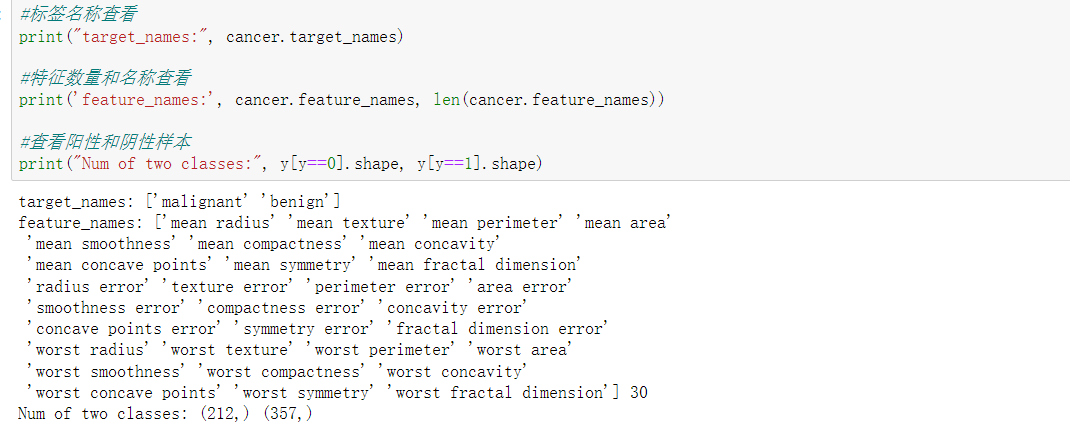
print(clf.predict(X\_test))

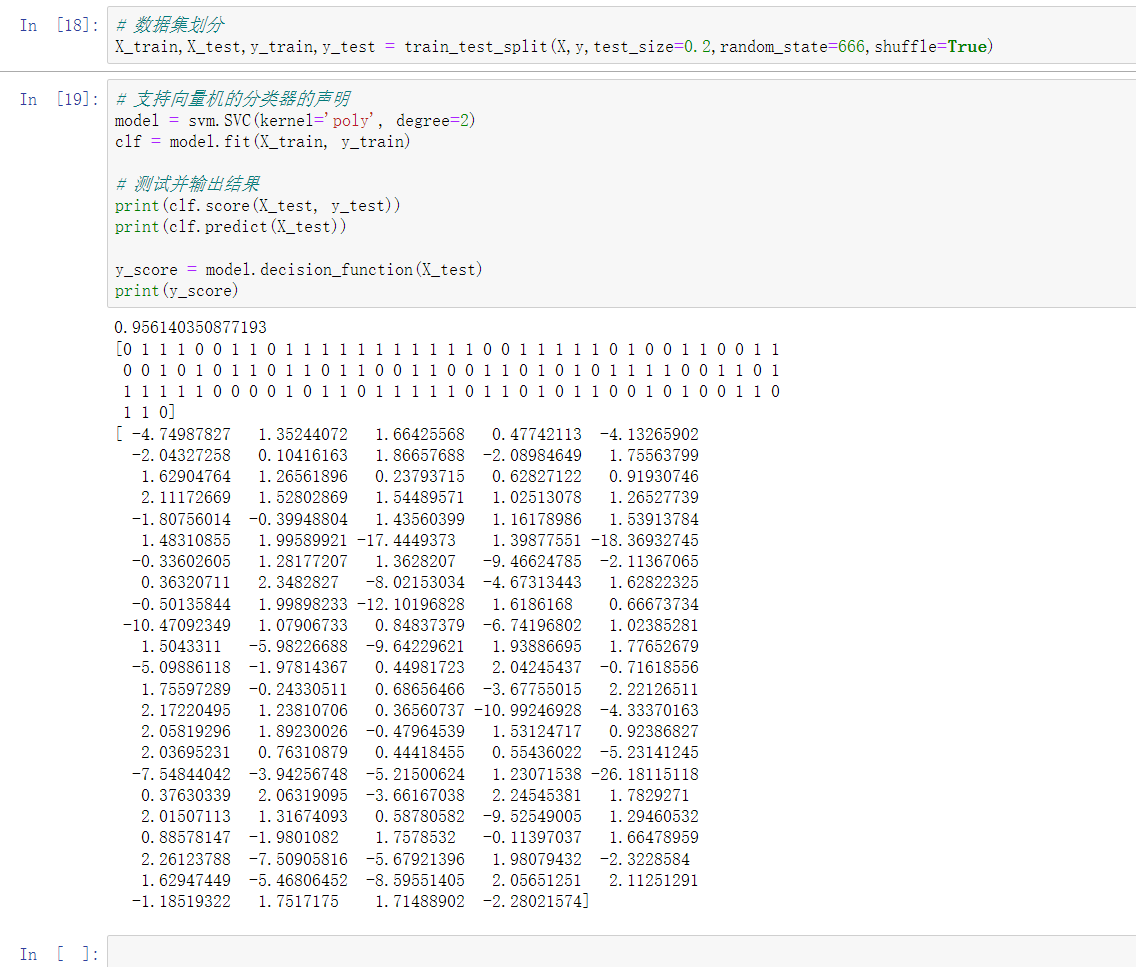
y\_score = model.decision\_function(X\_test)

print(y\_score)

结果及分析







题目

读取乳腺癌数据集并查看总体信息情况，然后对数据进行标准化处理，再然后进行数据集与测试集的划分，使用MLP多层感知器网络训练并测试，最后得到5折交叉验证的得分

代码

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPClassifier

# 读取数据

cancer = load\_breast\_cancer()

X,y = cancer.data, cancer.target

print("size of the data", X.shape, y.shape)

#标准化 x

scaler = StandardScaler()

X\_scaler = scaler.fit\_transform(X)

# 数据集划分

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X\_scaler,y,test\_size=0.2,random\_state=666,shuffle=True)

# 声明多层感知器网络

model = MLPClassifier(solver='lbfgs', activation='tanh', alpha=1e-5,

batch\_size='auto', beta\_1=0.9, beta\_2=0.999,

epsilon=1e-08, hidden\_layer\_sizes=(10,5),

learning\_rate="constant", learning\_rate\_init=0.001,

max\_iter=200,momentum=0.9)

# 训练与测试 MLP

clf = model.fit(X\_train, y\_train)

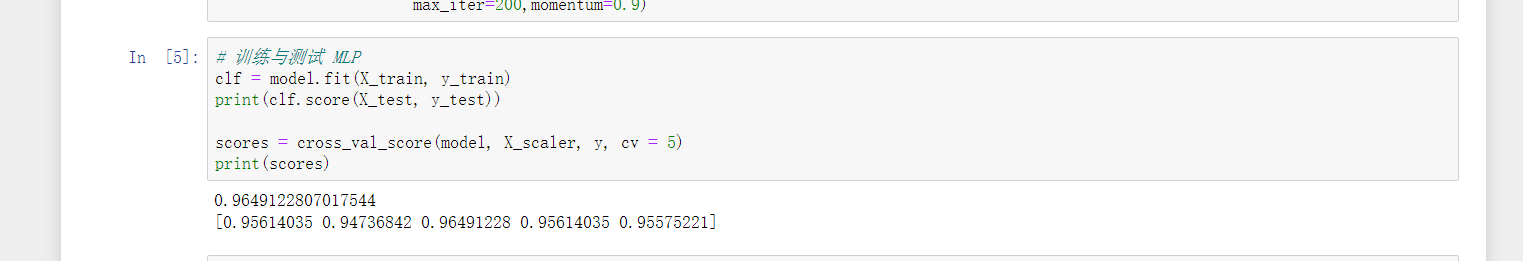
print(clf.score(X\_test, y\_test))

scores = cross\_val\_score(model, X\_scaler, y, cv = 5)

print(scores)

结果及分析





题目

导入手写数字的数据集，查看数据集的各项信息，展示部分手写数据的图片，使用PCA，ISOmap,LLE,t-SNE分别进行降维，最后评判每种方法的效果

代码

#coding:utf-8

from time import time

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d.axes3d import Axes3D

from sklearn import (datasets, decomposition,manifold)

from pylab import mpl

from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier as RFC

from sklearn.model\_selection import cross\_val\_score

from sklearn.neighbors import KNeighborsClassifier as KNN

mpl.rcParams['font.family'] = 'SimHei' ### # matplotlib其实是不支持显示中文的 显示中文需要一行代码设置字体

plt.rcParams['axes.unicode\_minus'] = False

#加载数据，显示数据

digits = datasets.load\_digits()

X = digits.data

y = digits.target

print (X.shape,y.shape)

n\_row = 20

img = np.zeros((15 \* n\_row, 15 \* n\_row))

for i in range(n\_row):

ix = 15 \* i + 1

for j in range(n\_row):

iy = 15 \* j + 1

img[ix:ix + 8, iy:iy + 8] = X[i \* n\_row + j].reshape((8, 8))

plt.imshow(img, cmap=plt.cm.binary)

plt.title('部分样本展示')

#%%

# 将降维后的数据可视化,2维

def plot\_embedding\_2d(X, title=None):

#坐标缩放到[0,1]区间

x\_min, x\_max = np.min(X,axis=0), np.max(X,axis=0)

X = (X - x\_min) / (x\_max - x\_min)

#降维后的坐标为（X[i, 0], X[i, 1]），在该位置画出对应的digits

fig = plt.figure()

ax = fig.add\_subplot(1, 1, 1)

for i in range(X.shape[0]):

ax.text(X[i, 0], X[i, 1],str(digits.target[i]),

color=plt.cm.Set1(y[i] / 10.),

fontdict={'weight': 'bold', 'size': 9})

if title is not None:

plt.title(title)

#%%

#将降维后的数据可视化,3维

def plot\_embedding\_3d(X, title=None):

#坐标缩放到[0,1]区间

x\_min, x\_max = np.min(X,axis=0), np.max(X,axis=0)

X = (X - x\_min) / (x\_max - x\_min)

#降维后的坐标为（X[i, 0], X[i, 1],X[i,2]），在该位置画出对应的digits

fig = plt.figure()

ax = fig.add\_subplot(1, 1, 1, projection='3d')

for i in range(X.shape[0]):

ax.text(X[i, 0], X[i, 1], X[i,2],str(digits.target[i]),

color=plt.cm.Set1(y[i] / 10.),

fontdict={'weight': 'bold', 'size': 9})

if title is not None:

plt.title(title)

plt.show()

# 默认的PCA

print(" PCA降维")

t0 = time()

X\_pca =decomposition.PCA(n\_components=3).fit\_transform(X)

# plot\_embedding\_2d(X\_pca[:,0:2],"PCA 2D")

# plot\_embedding\_3d(X\_pca,"PCA 3D (time %.2fs)" %(time() - t0))

# plt.show()

print(cross\_val\_score(RFC(n\_estimators=100,random\_state=0),X\_pca,y,cv=5).mean()) # 随机森林做交叉验证

# 累计方差贡献率曲线

pca\_line = PCA().fit(X)

plt.figure(figsize=[20,5])

plt.plot(np.cumsum(pca\_line.explained\_variance\_ratio\_))

plt.xlabel("number of components after dimension reduction")

plt.ylabel("cumulative explained variance ratio")

plt.show()

# 降维后维度的学习曲线,寻找最优参数

score = []

for i in range(1,40):

X\_dr = PCA(i).fit\_transform(X)

once = cross\_val\_score(RFC(n\_estimators=10,random\_state=0)

,X\_dr,y,cv=5).mean()

score.append(once)

plt.figure(figsize=[20,5])

plt.plot(range(1,40),score)

plt.show()

# 最终优化

X\_res\_pca = PCA(5).fit\_transform(X)

cross\_val\_score(RFC(n\_estimators=100,random\_state=0),X\_res\_pca,y,cv=5).mean() # 随机森林做交叉验证

# #%%

#Isomap

print("Isomap 降维")

t0 = time()

X\_iso = manifold.Isomap(n\_neighbors=10,n\_components=2).fit\_transform(X)

print("Done.")

plot\_embedding\_2d(X\_iso,"Isomap (time %.2fs)" %(time() - t0))

plt.show()

#standard LLE

print("LLE 降维")

clf = manifold.LocallyLinearEmbedding(n\_neighbors=10, n\_components=2,method='standard')

t0 = time()

X\_lle = clf.fit\_transform(X)

plot\_embedding\_2d(X\_lle,"Locally Linear Embedding (time %.2fs)" %(time() - t0))

plt.show()

# t-SNE

print(" t-SNE 降维")

tsne = manifold.TSNE(n\_components=3, init='pca', random\_state=0)

t0 = time()

X\_tsne = tsne.fit\_transform(X)

print (X\_tsne.shape)

plot\_embedding\_2d(X\_tsne[:,0:2],"t-SNE 2D")

plot\_embedding\_3d(X\_tsne,"t-SNE 3D (time %.2fs)" %(time() - t0))

plt.show()

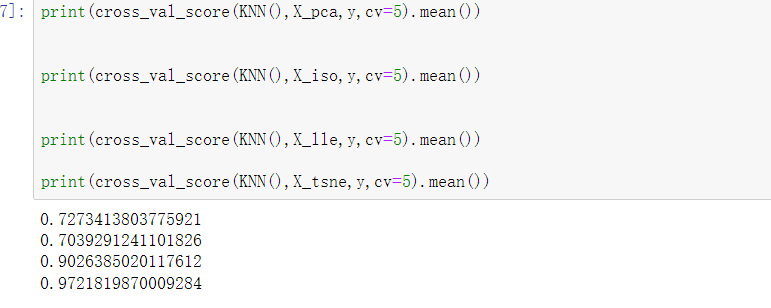
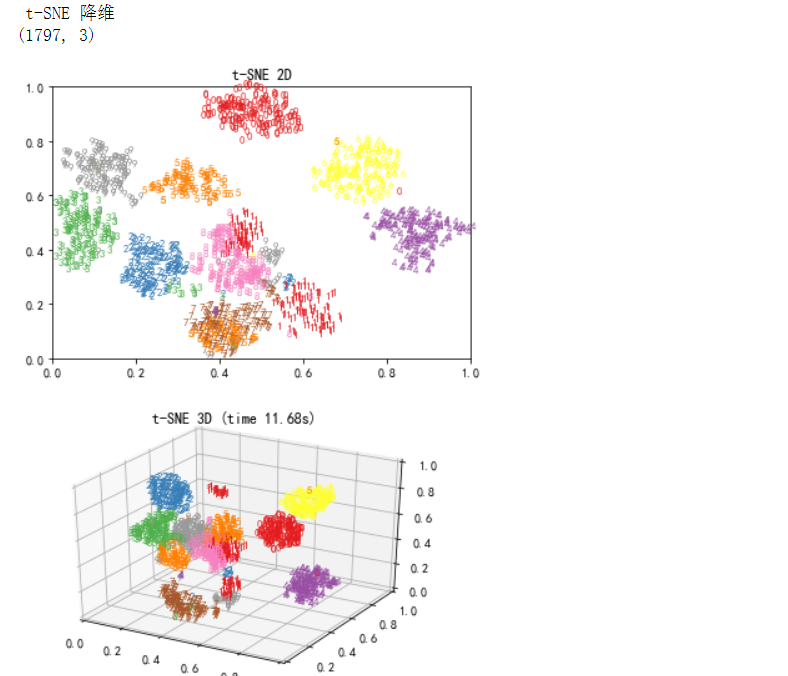
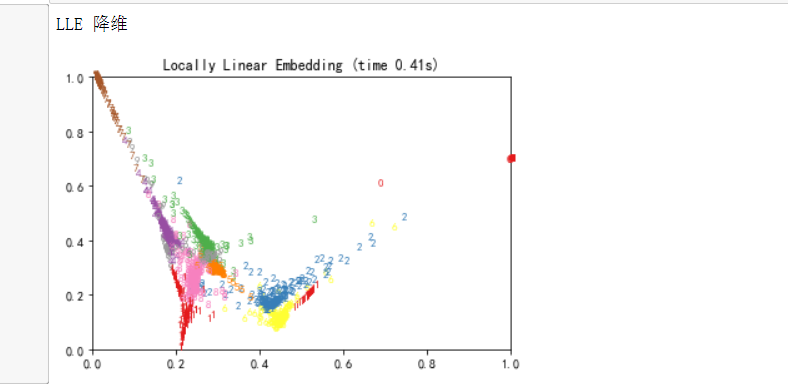
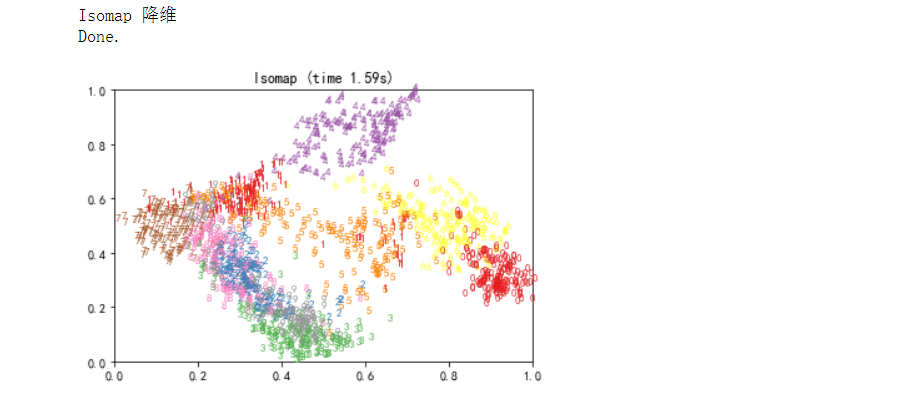
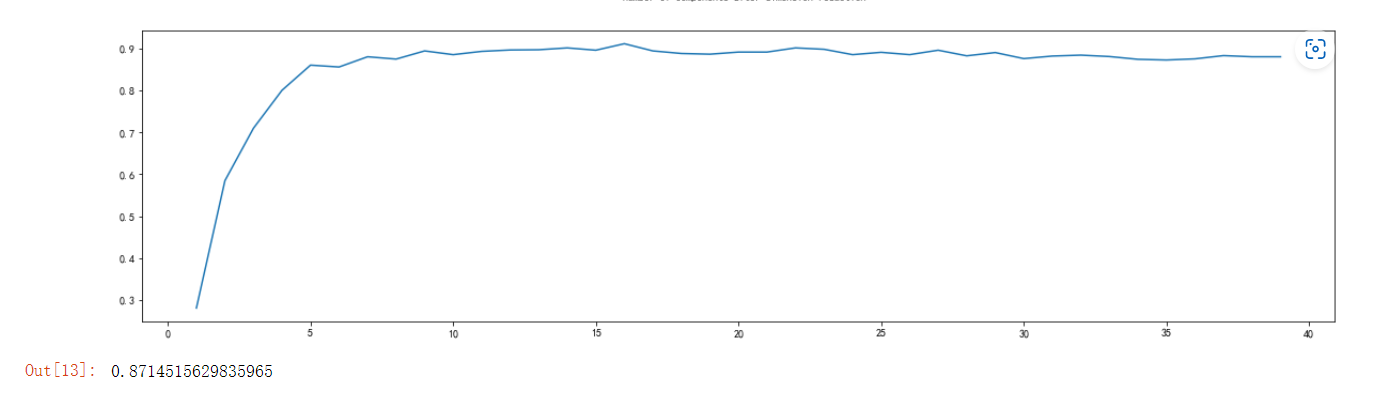
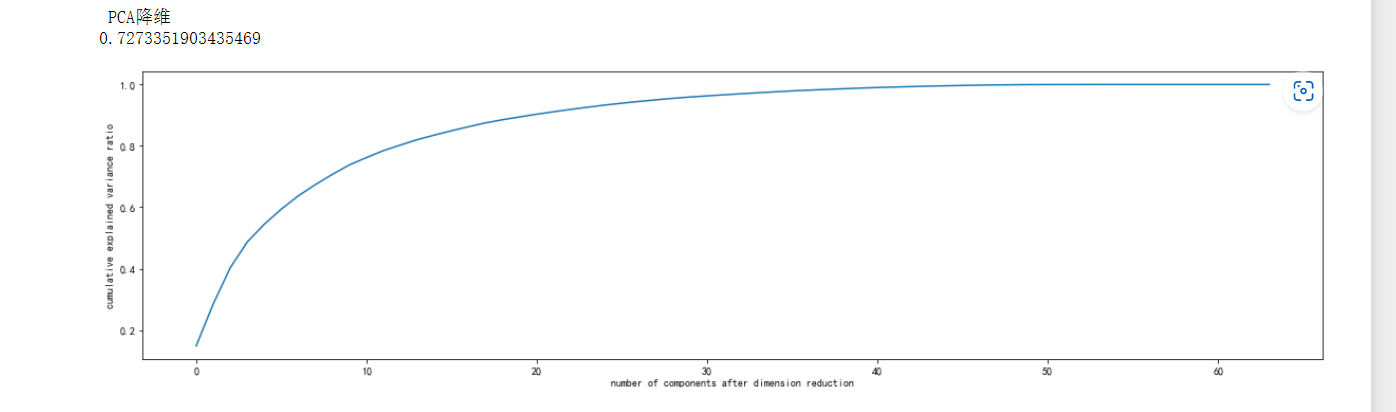
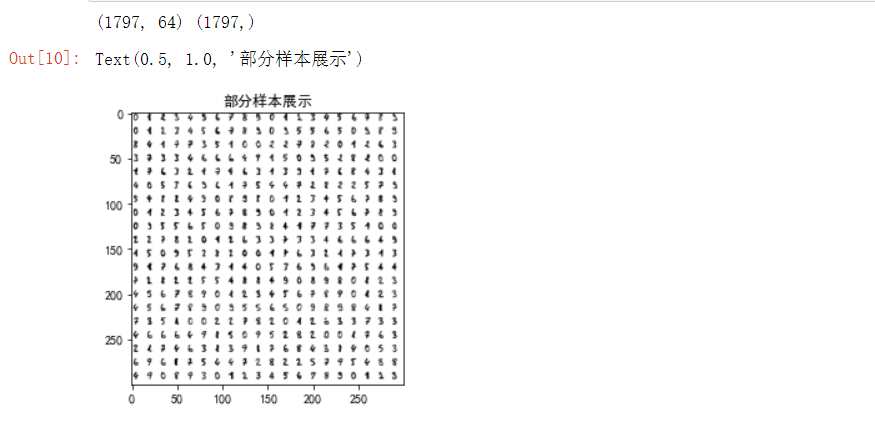
print(cross\_val\_score(KNN(),X\_pca,y,cv=5).mean())

print(cross\_val\_score(KNN(),X\_iso,y,cv=5).mean())

print(cross\_val\_score(KNN(),X\_lle,y,cv=5).mean())

print(cross\_val\_score(KNN(),X\_tsne,y,cv=5).mean())

结果及分析



使用默认的PCA进行降维，在随机森林上5折交叉验证的准确率为0.727，在画累计方差贡献率曲线，找最佳降维后维度的范围后，调整基评估器数量n\_estimators到学习曲线中最优的值，也能得到0.87的准确率

后续分别使用默认参数的其他方法实验降维，画出的降维2D和3D图效果都不错，最后使用KNN的5折交叉验证对每种方法进行了检验，准确率上从低到高的方法排名是isomap，pca，lle，t-SNE，由于使用的都是默认参数，实际进行超参数调参后效果会更佳，限于时间原因无法一一尝试，但在比knn效果更差的随机森林上调参的效果也是很显著的，证明别的方法也是有很大的优化空间的。