Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра комп’ютерного моделювання процесів і систем

ЗВІТ

з лабораторної роботи №10

“Зниження розмірності”

з курсу

«Алгоритми та моделі збору, аналізу та візуалізації даних»

Виконав: студент групи ІКМ-М222к Черкас Ю.В.

Перевірила: аспірантка Рикова В.О.

Харків 2023р

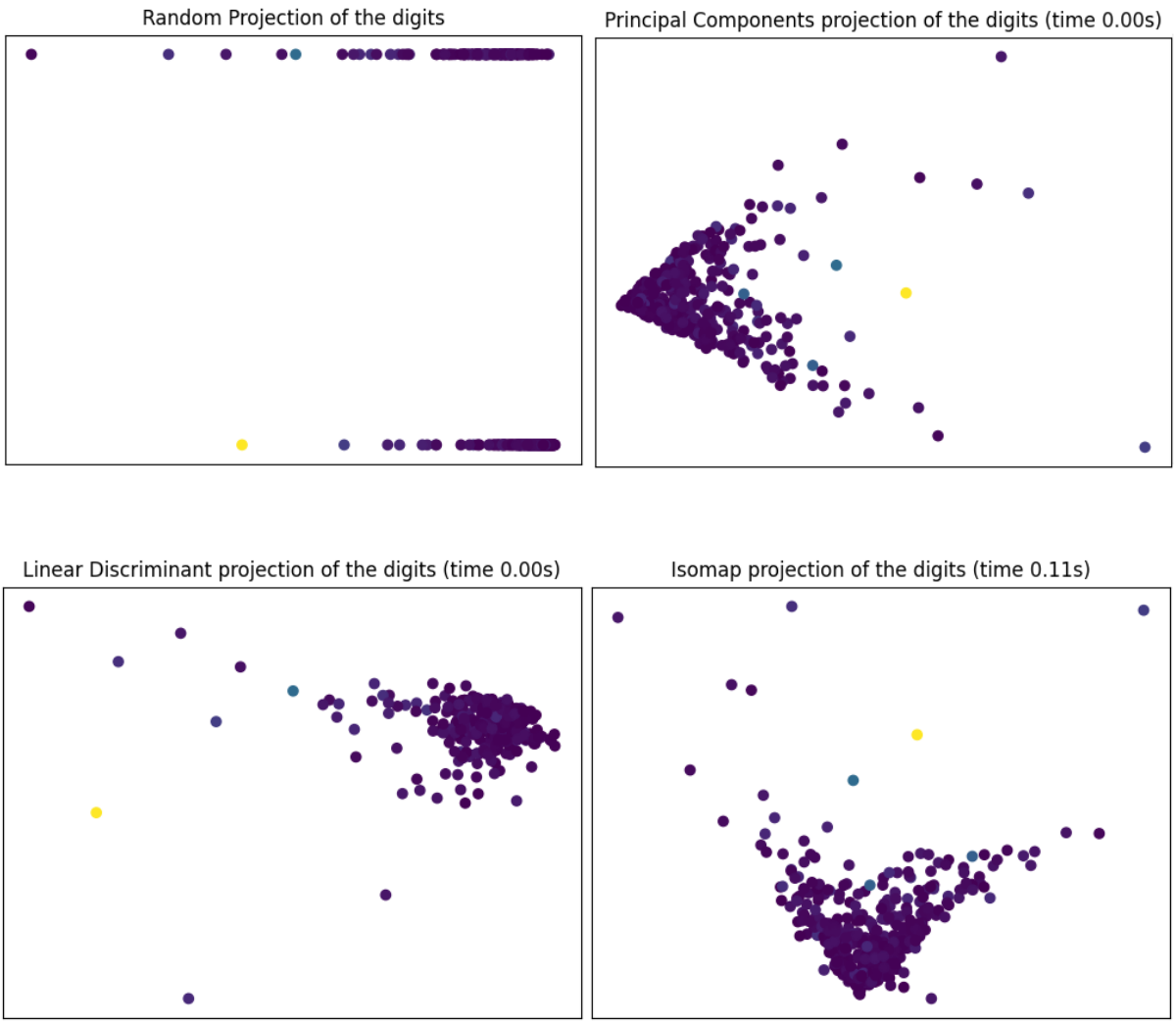
Варіант №15

Для відповідного датасету згідно з варіантом виконати пониження розмірності даних до просторів з розмірностями два та три (2D та 3D). Для пониження розмірності використовуйте всі доступні методи бібліотеки scikit-learn для зниження розмірності. Порівняйте результати та визначте яким методом було досягнуто найкращий результат.

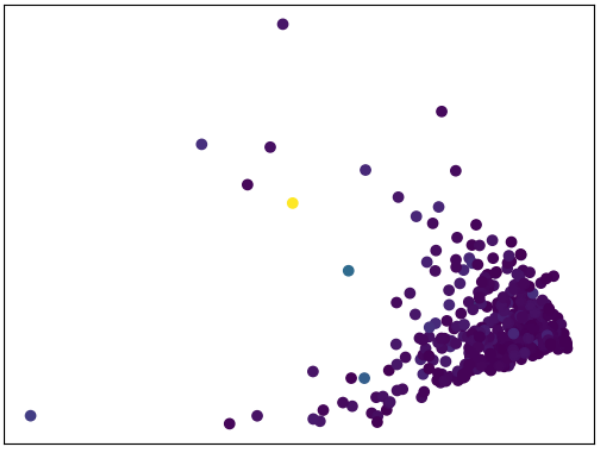
Виконання

Лістинг програми наведений в Додатку А.

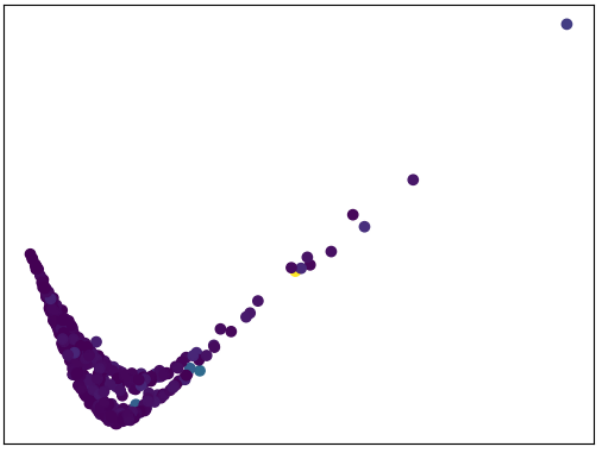
Графіки пониження розмірності 2D



Locally Linear Embedding of the digits (time 0.06s)



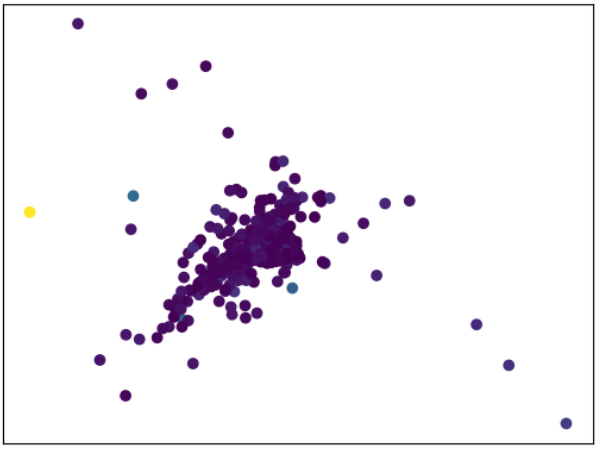
Modified Locally Linear Embedding of the digits (time 0.11s)



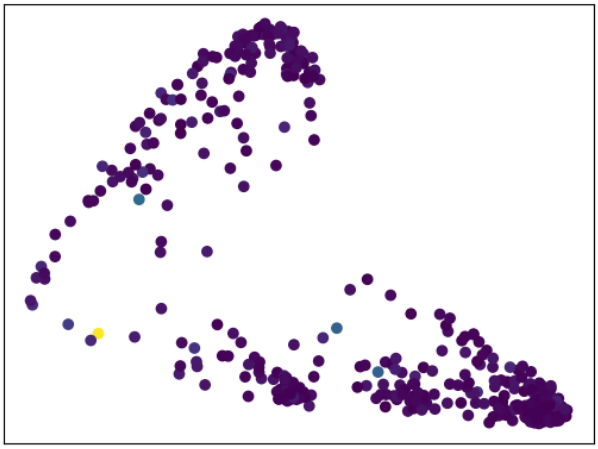
Hessian Locally Linear Embedding of the digits (time 0.11s)

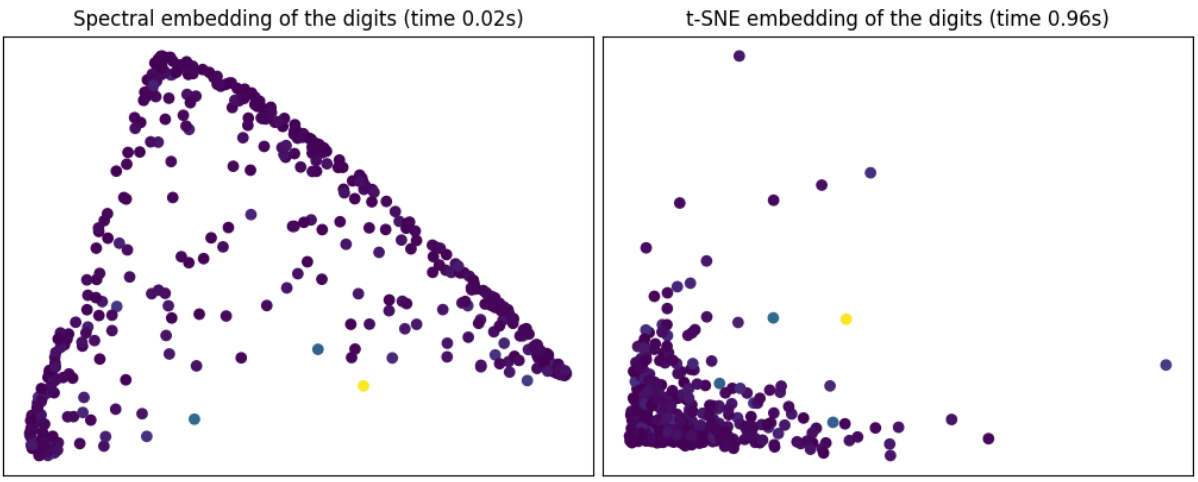
Local Tangent Space Alignment of the digits (time 0.09s)

MDS embedding of the digits (time 0.37s)

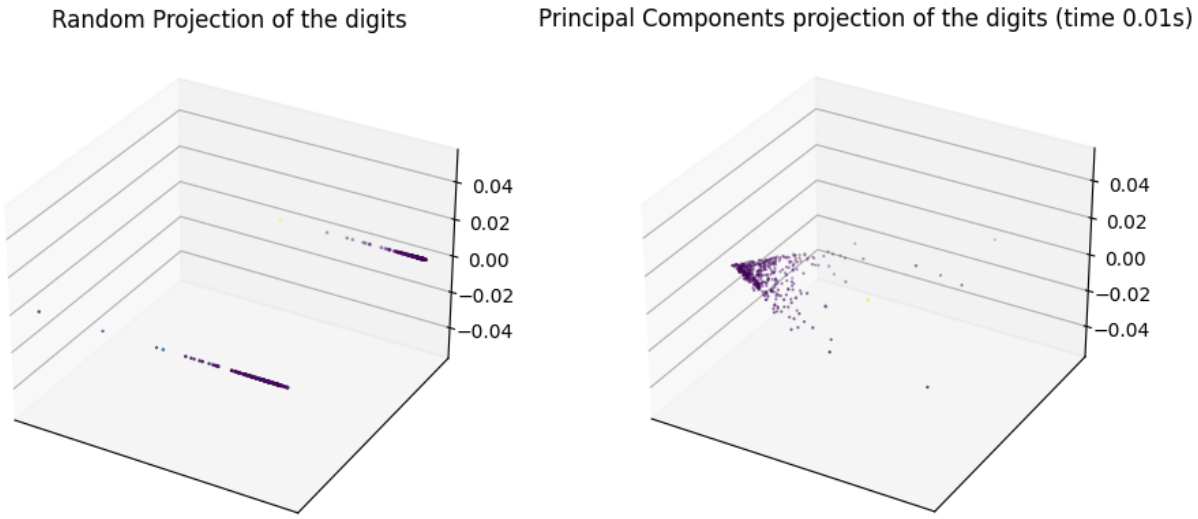


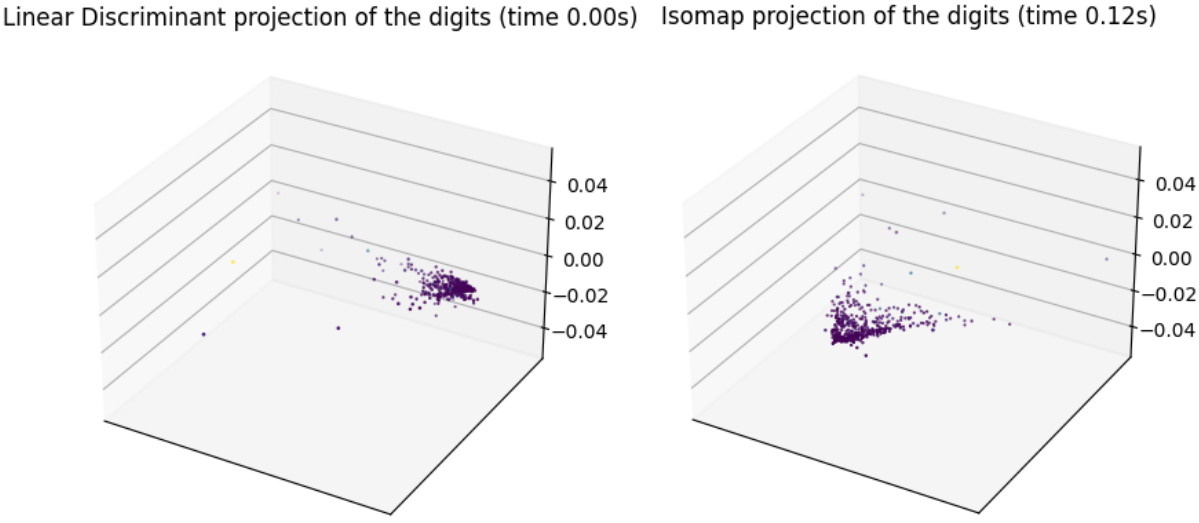
Random forest embedding of the digits (time 0.27s)



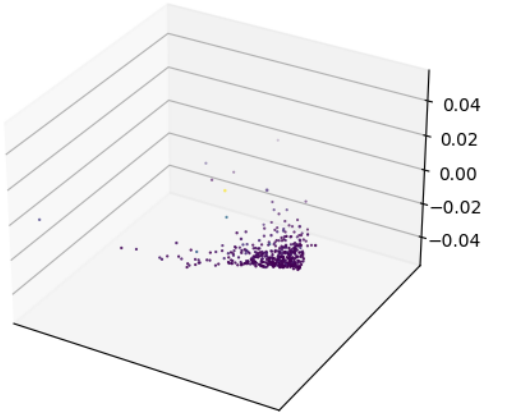


Графіки пониження розмірності 3D

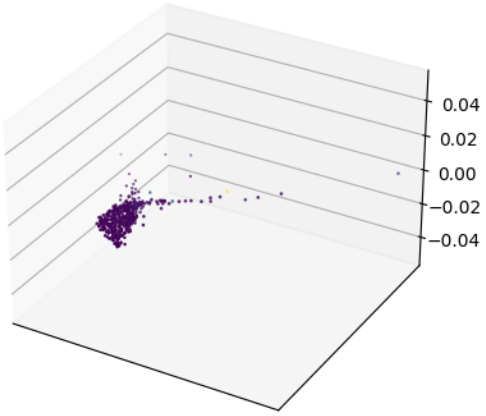




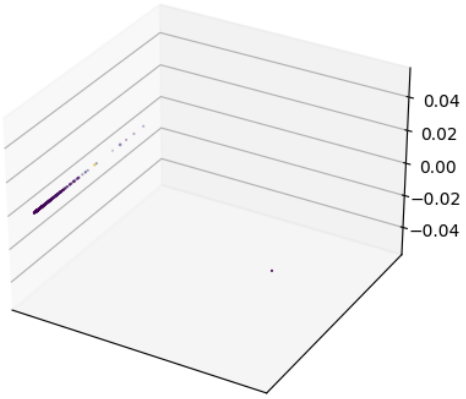
Locally Linear Embedding of the digits (time 0.07s)



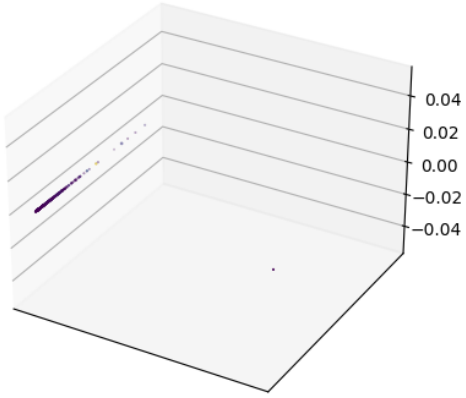
Modified Locally Linear Embedding of the digits (time 0.10s)



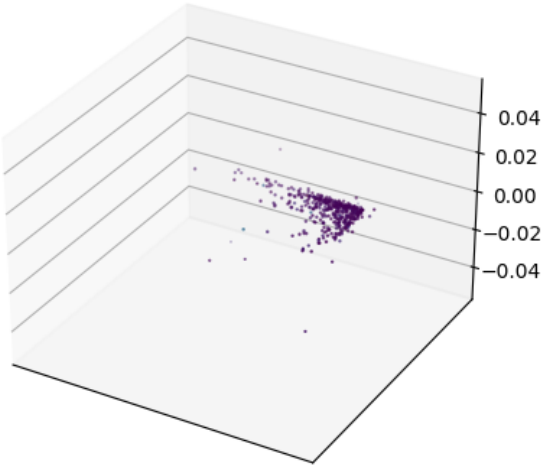
Hessian Locally Linear Embedding of the digits (time 0.12s)



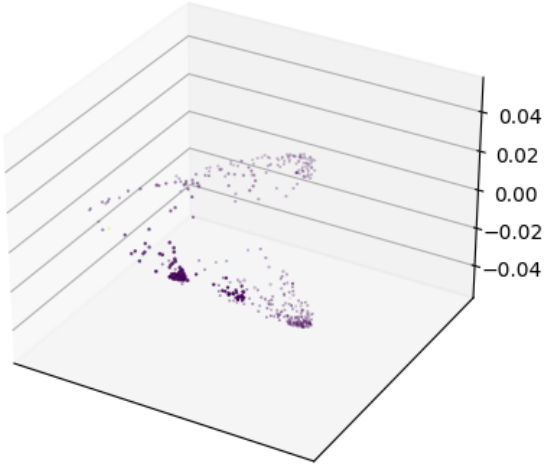
Local Tangent Space Alignment of the digits (time 0.09s)

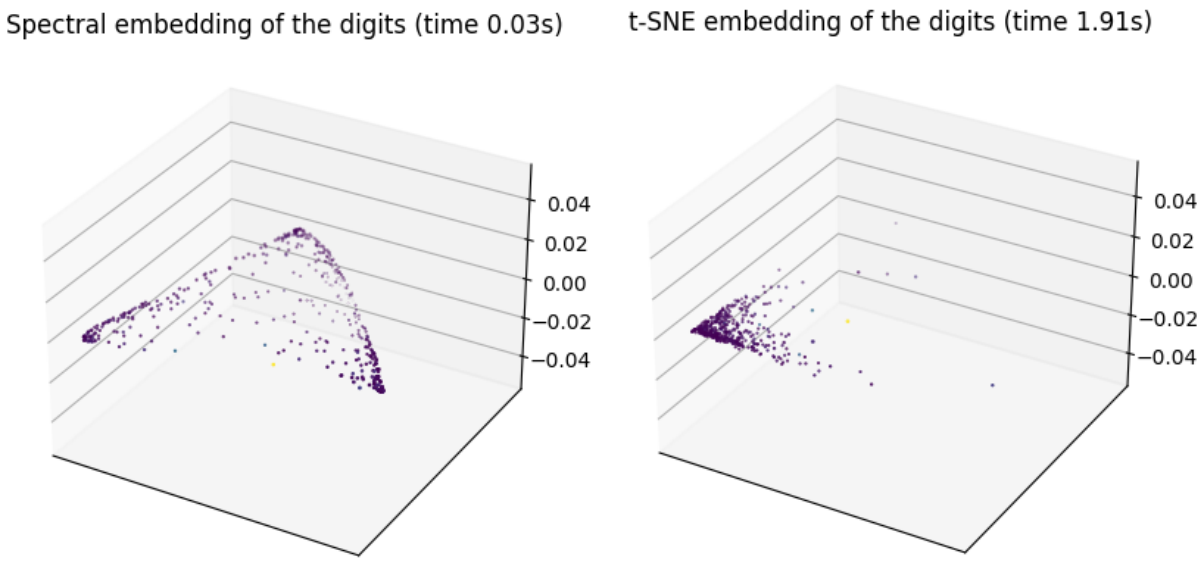


MDS embedding of the digits (time 0.36s)



Random forest embedding of the digits (time 0.27s)





Висновок: на даній лабораторній роботі ми дослідили різні методи пониження розмірності даних. Використання даних методів дозволяє більш ефективніше аналізувати дані високої розмірності. Побачили, що візуалізація одних і тих же даних кардинально відрізняється від методу до методу. Проте результати аналізу залежать не тільки від методів пониження розмірності, а і від характеру самих даних. Для даних аналізованих в даній роботі ('Wholesale customers data.csv' – дані оптових продаж), відсутня явно виражена категоризація. Проте даний факт не применшує можливість аналізу загальної структури вибірки даних в цілому.

Додаток А

Лістинг програми

```
# Authors: Fabian Pedregosa <fabian.pedregosa@inria.fr>
#          Olivier Grisel <olivier.grisel@ensta.org>
#          Mathieu Blondel <mathieu@mblondel.org>
#          Gael Varoquaux
# License: BSD 3 clause (C) INRIA 2011

print(__doc__)
from time import time

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble,
discriminant_analysis, random_projection)

digits = pd.read_csv('Wholesale customers data.csv')
X = digits.iloc[:,:-1].values.astype(np.float)
y = digits.iloc[:,-1].values
n_samples, n_features = X.shape
n_neighbors = 30

#----------------------------------------------------------------------
# Scale and visualize the embedding vectors
def plot_embedding(X, title=None):
    x_min, x_max = np.min(X, 0), np.max(X, 0)
    X = (X - x_min) / (x_max - x_min)

    plt.figure()
    ax = plt.subplot(111)

    plt.scatter(X[:,0], X[:,1], c = y)

    plt.xticks([]), plt.yticks([])
    if title is not None:
        plt.title(title)


#----------------------------------------------------------------------
# Random 2D projection using a random unitary matrix
print("Computing random projection")
rp = random_projection.SparseRandomProjection(n_components=2, random_state=42)
X_projected = rp.fit_transform(X)
plot_embedding(X_projected, "Random Projection of the digits")
```

```
#----------------------------------------------------------------------
# Projection on to the first 2 principal components

print("Computing PCA projection")
t0 = time()
X_pca = decomposition.TruncatedSVD(n_components=2).fit_transform(X)
plot_embedding(X_pca,
               "Principal Components projection of the digits (time %.2fs)" %
               (time() - t0))

#----------------------------------------------------------------------
# Projection on to the first 2 linear discriminant components

print("Computing Linear Discriminant Analysis projection")
X2 = X.copy()
X2.flat[::X.shape[1] + 1] += 0.01  # Make X invertible
t0 = time()
X_lda =
discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform
(X2, y)
plot_embedding(X_lda,
               "Linear Discriminant projection of the digits (time %.2fs)" %
               (time() - t0))


#----------------------------------------------------------------------
# Isomap projection of the digits dataset
print("Computing Isomap embedding")
t0 = time()
X_iso = manifold.Isomap(n_neighbors = n_neighbors,
n_components=2).fit_transform(X)
print("Done.")
plot_embedding(X_iso,
               "Isomap projection of the digits (time %.2fs)" %
               (time() - t0))


#----------------------------------------------------------------------
# Locally linear embedding of the digits dataset
print("Computing LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors = n_neighbors,
n_components=2,
                                      method='standard')
t0 = time()
X_lle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_lle,
```

```
               "Locally Linear Embedding of the digits (time %.2fs)" %
               (time() - t0))


#----------------------------------------------------------------------
# Modified Locally linear embedding of the digits dataset
print("Computing modified LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors = n_neighbors,
n_components=2,
                                      method='modified')
t0 = time()
X_mlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_mlle,
               "Modified Locally Linear Embedding of the digits (time %.2fs)"
%
               (time() - t0))


#----------------------------------------------------------------------
# HLLE embedding of the digits dataset
print("Computing Hessian LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors = n_neighbors,
n_components=2,
                                      method='hessian', eigen_solver =
'dense')
t0 = time()
X_hlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_hlle,
               "Hessian Locally Linear Embedding of the digits (time %.2fs)" %
               (time() - t0))


#----------------------------------------------------------------------
# LTSA embedding of the digits dataset
print("Computing LTSA embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors = n_neighbors,
n_components=2,
                                      method='ltsa', eigen_solver = 'dense')
t0 = time()
X_ltsa = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_ltsa,
               "Local Tangent Space Alignment of the digits (time %.2fs)" %
               (time() - t0))

#----------------------------------------------------------------------
# MDS  embedding of the digits dataset
```

```
print("Computing MDS embedding")
clf = manifold.MDS(n_components=2, n_init=1, max_iter=100)
t0 = time()
X_mds = clf.fit_transform(X)
print("Done. Stress: %f" % clf.stress_)
plot_embedding(X_mds,
               "MDS embedding of the digits (time %.2fs)" %
               (time() - t0))

#----------------------------------------------------------------------
# Random Trees embedding of the digits dataset
print("Computing Totally Random Trees embedding")
hasher = ensemble.RandomTreesEmbedding(n_estimators=200, random_state=0,
                                       max_depth=5)
t0 = time()
X_transformed = hasher.fit_transform(X)
pca = decomposition.TruncatedSVD(n_components=2)
X_reduced = pca.fit_transform(X_transformed)

plot_embedding(X_reduced,
               "Random forest embedding of the digits (time %.2fs)" %
               (time() - t0))

#----------------------------------------------------------------------
# Spectral embedding of the digits dataset
print("Computing Spectral embedding")
embedder = manifold.SpectralEmbedding(n_components=2, random_state=0,
                                      eigen_solver="arpack")
t0 = time()
X_se = embedder.fit_transform(X)

plot_embedding(X_se,
               "Spectral embedding of the digits (time %.2fs)" %
               (time() - t0))

#----------------------------------------------------------------------
# t-SNE embedding of the digits dataset
print("Computing t-SNE embedding")
tsne = manifold.TSNE(n_components=2, init='pca', random_state=0)
t0 = time()
X_tsne = tsne.fit_transform(X)

plot_embedding(X_tsne,
               "t-SNE embedding of the digits (time %.2fs)" %
               (time() - t0))

plt.show()
```