Ex4: SVD Dimensionality Reduction

Câu 1:

- Cho tập tin iris.csv, đọc dữ liệu ra dataframe và chỉ lấy 4 cột đầu
- Từ datafarme, sử dụng SVD bằng công thức để giảm chiều dữ liệu chỉ còn 2 component
- Trực quan hóa dữ liệu sau khi giảm chiều, có cả cột species

Câu 2:

- Tải dữ liệu digits từ dataset của sklearn
- Từ dữ liệu, sử dụng TruncatedSVD để giảm chiều dữ liệu chỉ còn 10 component

Câu 1: Gợi ý

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [2]:
```
iris = pd.read_csv("iris.csv")
iris.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [3]:
```
X = iris[['sepal_length','sepal_width','petal_length','petal_width']]
X.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [4]:
```
# SVD
U, s, VT = np.linalg.svd(X)
```

In [5]:
```
U
```

Out[5]:
```
array([[-6.16171172e-02,  1.29969428e-01, -5.58364155e-05, ...,
        -9.34637342e-02, -9.60224157e-02, -8.09922905e-02],
       [-5.80722977e-02,  1.11371452e-01,  6.84386629e-02, ...,
         3.66755322e-02, -3.24463474e-02,  1.27273399e-02],
       [-5.67633852e-02,  1.18294769e-01,  2.31062793e-03, ...,
         3.08252776e-02,  1.95234663e-01,  1.35567696e-01],
       ...,
       [-9.40702260e-02, -4.98348018e-02, -4.14958083e-02, ...,
         9.81822841e-01, -2.17978813e-02, -8.85972146e-03],
       [-9.48993908e-02, -5.62107520e-02, -2.12386574e-01, ...,
        -2.14264126e-02,  9.42038920e-01, -2.96933496e-02],
       [-8.84882764e-02, -5.16210172e-02, -9.51442925e-02, ...,
        -8.52768485e-03, -3.02139863e-02,  9.73577349e-01]])
```

In [6]:
```
s
```

Out[6]:
```
array([95.95066751, 17.72295328,  3.46929666,  1.87891236])
```

In [7]:
```
VT
```

Out[7]:
```
array([[-0.75116805, -0.37978837, -0.51315094, -0.16787934],
       [ 0.28583096,  0.54488976, -0.70889874, -0.34475845],
       [ 0.49942378, -0.67502499, -0.05471983, -0.54029889],
       [ 0.32345496, -0.32124324, -0.48077482,  0.74902286]])
```

In [8]:
```
Sigma = np.zeros(X.shape)
Sigma[:X.shape[1],:X.shape[1]] = np.diag(s)
Sigma[0:5]
```

Out[8]:
```
array([[95.95066751,  0.        ,  0.        ,  0.        ],
       [ 0.        , 17.72295328,  0.        ,  0.        ],
       [ 0.        ,  0.        ,  3.46929666,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  1.87891236],
       [ 0.        ,  0.        ,  0.        ,  0.        ]])
```

In [9]:
```
n_componets = 2
Sigma = Sigma[:,:n_componets]
VT = VT[:n_componets, :]
```

In [10]:
```
# Solution 1
T_s1 = U.dot(Sigma)
```

In [11]:
```
T_s1[0:5]
```

Out[11]:
```
array([[-5.91220352,  2.30344211],
       [-5.57207573,  1.97383104],
       [-5.4464847 ,  2.09653267],
       [-5.43601924,  1.87168085],
       [-5.87506555,  2.32934799]])
```

In [12]:
```
T_s2 = X.dot(VT.T)
```

In [13]:
```
T_s2[0:5]
```

Out[13]:

	0	1
0	-5.912204	2.303442
1	-5.572076	1.973831
2	-5.446485	2.096533
3	-5.436019	1.871681
4	-5.875066	2.329348

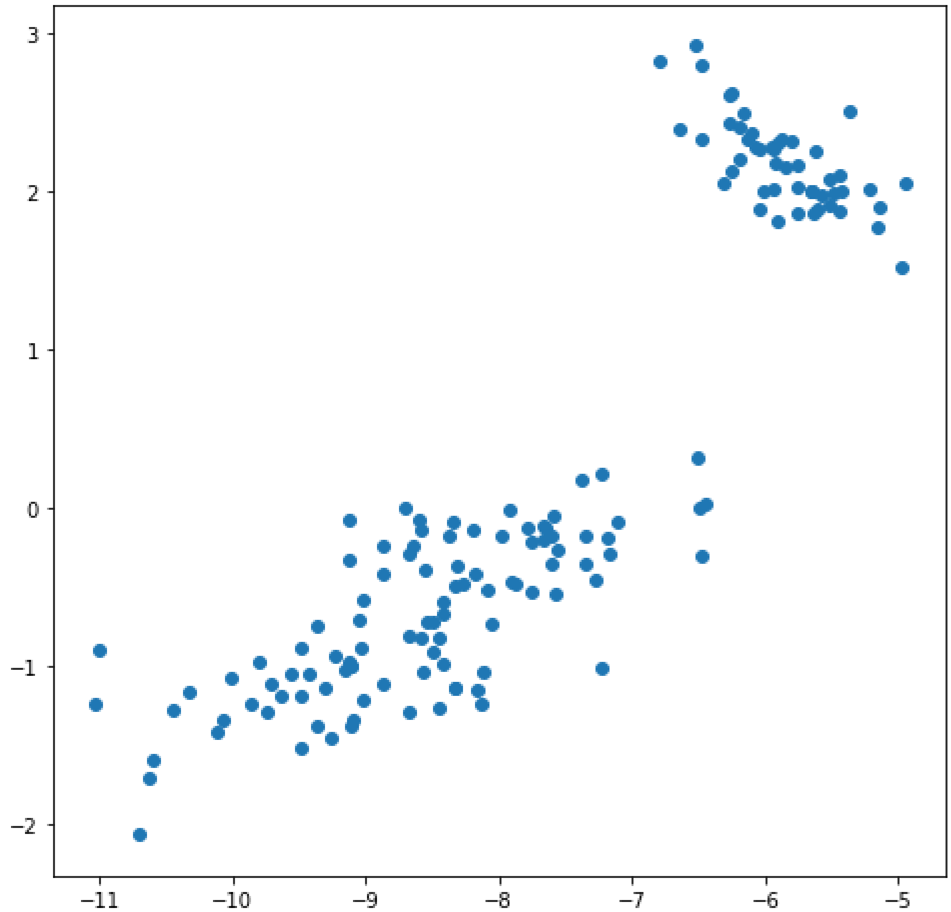
In [14]:
```
T_s2.columns = ["comp1", "comp2"]
```

In [15]:
```
T_s2["species"] = iris['species']
T_s2.head()
```

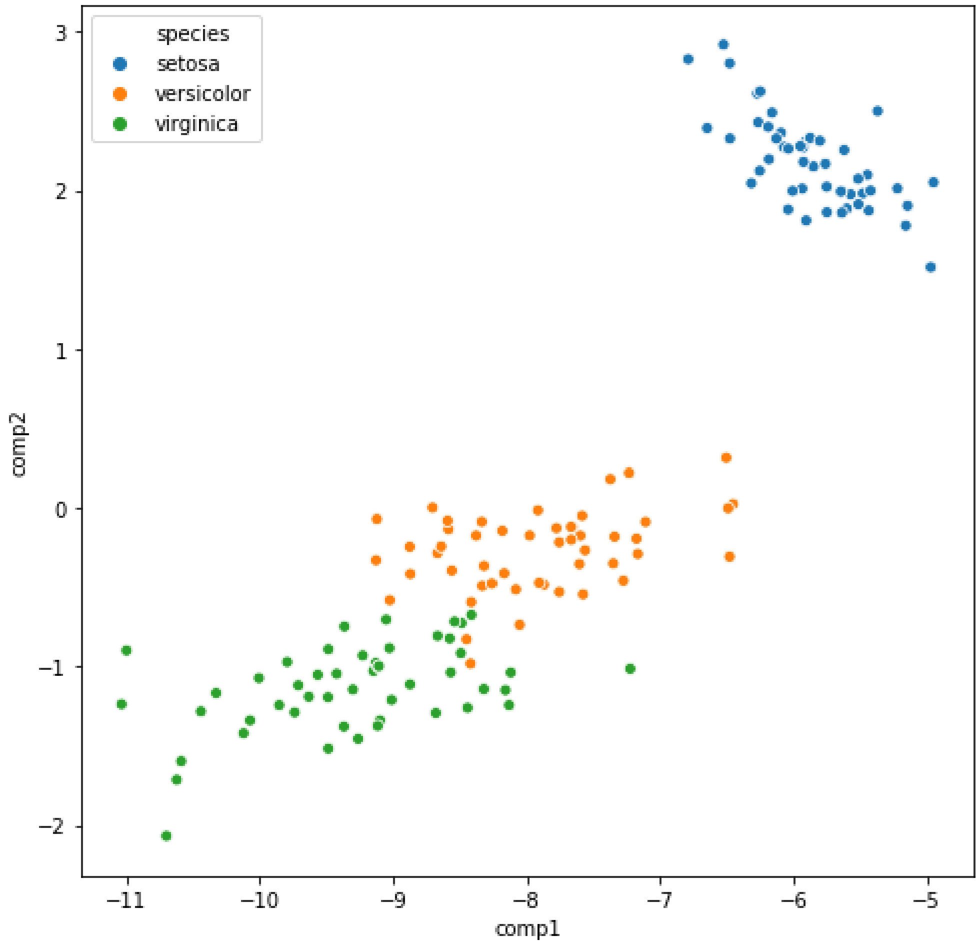
Out[15]:

	comp1	comp2	species
0	-5.912204	2.303442	setosa
1	-5.572076	1.973831	setosa
2	-5.446485	2.096533	setosa
3	-5.436019	1.871681	setosa
4	-5.875066	2.329348	setosa

In [16]:
```
plt.figure(figsize=(8,8))
plt.scatter(T_s2["comp1"], T_s2["comp2"])
plt.show()
```



In [17]:
```
plt.figure(figsize=(8,8))
sns.scatterplot(x="comp1", y="comp2",data=T_s2, hue="species")
plt.show()
```



Câu 2: Gợi ý

- Tải dữ liệu digits từ dataset của sklearn
- Từ dữ liệu, sử dụng TruncatedSVD để giảm chiều dữ liệu chỉ còn 10 component
- Trực quan hóa dữ liệu sau khi giảm chiều

In [18]:
```
# Load libraries
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import TruncatedSVD
from scipy.sparse import csr_matrix
from sklearn import datasets
import numpy as np
```

In [19]:
```
# Load the data
digits = datasets.load_digits()
print(digits.data)
```

```
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
```

In [20]:
```
digits.data.shape
```

Out[20]:
```
(1797, 64)
```

In [21]:
```
# Standardize the feature matrix
X = digits.data

# Make sparse matrix
X_sparse = csr_matrix(X)
```

In [22]:
```
X_sparse
```

Out[22]:
```
<1797x64 sparse matrix of type '<class 'numpy.float64'>'
	with 58736 stored elements in Compressed Sparse Row format>
```

In [23]:
```
# Create a TSVD
tsvd = TruncatedSVD(n_components=10)
```

In [24]:
```
# Conduct TSVD on sparse matrix
X_sparse_tsvd = tsvd.fit(X_sparse).transform(X_sparse)
```

In [25]:
```
# Show results
print('Original number of features:', X_sparse.shape[1])
print('Reduced number of features:', X_sparse_tsvd.shape[1])
```

```
Original number of features: 64
Reduced number of features: 10
```

In [26]:
```
X[0]
```

Out[26]:
```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

In [27]:
```
X_sparse_tsvd[0]
```

Out[27]:
```
array([45.86127719, -1.1921157 , 21.10005928, -9.48896858, 13.04312615,
       -7.01647258, -8.96014232,  0.39659636,  1.31015904, -1.38988172])
```

In [28]:
```
# View Percent Of Variance Explained By New Features
# Sum of 10 components' explained variance ratios
tsvd.explained_variance_ratio_[0:10].sum()
```

Out[28]:
```
0.7324264436815388
```

In [29]:
```
# 73% with 10 components
```