Labwork: data visualization

Objective
Perform data exploration and visualization using methods of dimension reduction such as PCA and t-SNE.

1 Starter: athlete's dataset

We consider the dataset `ais.csv` (see Moodle) consisting of physiological records (hemoglobin, body fat mass, ferritin level) of several practitioners (female and male) of different sports (rowing, swimming, tennis). The task to undertake is to explore correlation between the variables (physiological measurements) and to visualize the athletes' 2d representation.

1. Load the data and show its summary. How many samples and inputs do we have?

```
import pandas as pd
import numpy as np

dataset = "ais.csv"
df = pd.read_csv(dataset, delimiter=";")
print("Variable names")
print(df.columns)

# cardinality of each sport group
a, b = np.unique(df["sport"], return_counts=True)
for sp, card in zip(a,b):
print(card, "practitioners of ", sp)

# drop some of the columns
drop_var = ["athlete", "sport"]
df = pd.DataFrame(df.drop(drop_var, axis=1))
```

2. Perform a statistical analysis of the data. Show the boxplots of each variable, the correlation matrix and comment on the plots.

```
import seaborn as sns
# boxplot of the input variables and the output
plt.figure(figsize=(12, 3))
variables = ["rcc", "wcc", "hc", "hg", "ferr", "bmi", "ssf", "pcBfat", "
    lbm", "ht", "wt"]
sns.boxplot(data=df[variables], orient="v")
#plt.xticks(fontsize=16), plt.yticks(fontsize=16)
plt.show()

#Correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, fmt=".2f")
plt.show()
```

3. For visualization sake, let consider the gender as the label of each athlete. We form accordingly the input matrix X and the labels Y.

```
X = df[variables].values
Y = df["sex"].values
```

4. Normalize the data X ()

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler(with_mean=True, with_std=True)
# to be filled up
...
# normalize X
X = ...
```

5. Perform PCA on the normalized data

```
from sklearn.decomposition import PCA

pca = PCA()
pca.fit(X)
```

6. Plot the eigen-spectrum (the explained variance of each projection axis) of the correlation matrix using the provided function `plot_spectre_variance_expl(valprop)`.

```
 ## trace du spectre des valeurs propres et de la variance expliquee
 def plot_spectre_variance_expl(valprop):
   tot = sum(np.abs(valprop))
   var_exp = [(i / tot) for i in sorted(np.abs(valprop), reverse=True)]
   cum_var_exp = np.cumsum(var_exp)

   plt.figure(figsize=(12, 6))
   plt.subplot(1, 2, 1)
   plt.bar(range(1, valprop.size + 1), np.abs(valprop), alpha=0.5, align
       ="center", label="Individual")
   plt.title("Correlation matrix Eigenvalues")
   plt.ylabel("Explained variance")
   plt.xlabel("Principal components")

   plt.subplot(1, 2, 2)
   markerline, stemlines, baseline = plt.stem(range(1, valprop.size + 1)
       , cum_var_exp, "-r", label="Individual")
   plt.setp(stemlines, "color", "r", "linewidth", 2)
   plt.plot(range(1, valprop.size + 1), cum_var_exp, "b-.",label="
       Cumulative")
   plt.title("Cumulated explained variance (in %)")
   plt.ylabel("Explained variance")
   plt.xlabel("Principal components")
   plt.tight_layout()
```

```
# Plot the spectrum
plot_spectre_variance_expl(pca.explained_variance_ratio_)
```

According to the eigen-spectrum how many principal components are required to well summarize the data ?

7. Using the learned PCA model, project the data in two-dimension (using two components). Visualize the data. Discuss the obtained scatter plot.

```
# Projection
n_axes = 2
pca.n_components = n_axes
X_proj_ais = pca.transform(X)

# 2d visualization
Y[Y=="f"] = 1
Y[Y=="m"]=0
plt.figure(figsize=(12,8))
plt.scatter(X_proj_ais[:,0], X_proj_ais[:, 1], c=Y, cmap=plt.cm.Set1,
    edgecolor="k")
plt.colorbar()
```

8. Repeat the last questions by applying PCA without normalizing the data X. Compare to the case where X is normalized. Justify the observed differences in the results?

2 Visualization of digits

In this section we will consider handwritten digits (from 0 to 9). Each digit consists of a 16×16 gray-level image. It is reshaped as a vector of dimension 256, stacked in the matrix X. The digits are corresponding labels (0-9) are stored in the file `uspsasi.mat` (see Moodle). The goal is twofold : (i) visualize in 2d the digits, and (ii) experiment the digit reconstruction after PCA.

1. Load the data and visualize some of the digits as an image.

```
import scipy.io as sio

# Load data
digits = sio.loadmat("uspsasi.mat")
digits, labels = digits["x"], digits["y"][:,0]

# Plot some digits
fig, ax_array = plt.subplots(6, 6)
axes = ax_array.flatten()
for i, ax in enumerate(axes):
  ax.imshow(digits[i].reshape(16, 16), cmap="gray_r")
plt.setp(axes, xticks=[], yticks=[], frame_on=False)
plt.tight_layout(h_pad=0.5, w_pad=0.01)
```

   Check the dimensionality (number of images and their size) of the dataset

2. Perform PCA on the digits and plot the explained variance curve. Discuss the number of principal components to retain.

3. Projected samples using PCA can be approximately reconstructed by applying the inverse transformation of PCA (see the lecture). We visually examine how good is the reconstruction when varying the number of retained principal components. Comments on the reconstruction quality.

```
n_axes_vect = np.array([2, 5, 10, 25, 50, 100, 200, 250])
idx = 1
plt.figure(figsize=(12, 12))
plt.subplot(1, n_axes_vect.size+2, 1),
plt.imshow(digits[idx].reshape(16, 16), cmap="gray_r")
plt.title("orginal")

for idx_k, n_axes in enumerate(n_axes_vect):
# fix the number of components
pca_digit = PCA(n_components=n_axes)

# fit the PCA model
pca_digit.fit(digits)

# projection
Xproj = pca_digit.transform(digits)

# reconstruction
X_hat = pca_digit.inverse_transform(Xproj)

plt.subplot(1, n_axes_vect.size+2, idx_k+2)
plt.imshow(X_hat[idx].reshape(16, 16), cmap="gray_r")
plt.title(n_axes)
```

4. Finally we want to visualize the data in 2d. Project the digits in 2d via PCA and show their scatter plot. Use the labels to distinguish the digit classes. Does PCA provide a good separation of the classes?

3 A non-linear projection method: t-SNE

Finally, we compare PCA with t-SNE in terms of data visualization.

```
from sklearn.manifold import TSNE as tsne

# define and apply t-sne
tsne = tsne(n_components=2)
X_proj_2d_tsne = tsne.fit_transform(digits)


# Visualization
plt.figure(figsize=(12, 7))
plt.scatter(X_proj_2d_tsne[:,0], X_proj_2d_tsne[:, 1], c=labels, cmap=plt.cm.
    Set1, edgecolor="k")
plt.colorbar()
plt.title("t-SNE")
```

Discuss the obtained results.