

IFT870/BIN710 – Corrigé – Exercice : Réduction de dimensions et Clustering

|| Aïda Ouangraoua (Dpt. Informatique, Fac. Sciences, Université de Sherbrooke)

Dans le cadre de cet exercice, nous procéderons à l'importation du jeu de données prédéfini « Iris ». Le lien de la documentation se trouve ci-après :

- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris

Question 1 : Jeu de données

- Charger le jeu de données à partir du dépôt scikit-learn.
- Charger le jeu de données à partir du fichier `iris_data.csv`.
- fournir une description détaillée du jeu de données, incluant son format, les types des attributs ainsi qu'une explication de chacun des attributs.

Corrigé Question 1

```
In [1]: # Charger le jeu de données à partir du dépôt scikit-learn.
from sklearn.datasets import load_iris
iris_dataset = load_iris()

In [2]: # Charger le jeu de données à partir du fichier <em>iris_data.csv</em>.
import pandas as pd
iris_dataset_pandas = pd.read_csv('./iris_data.csv', sep=';', index_col=0)
iris_dataset_pandas.head(5)

Out[2]:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	classe
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

```
In [3]: # Fournir une description détaillée du jeu de données, incluant son format, les types des attributs ainsi qu'une explication de chacun des attributs.
print("Data shape: ", iris_dataset_pandas.shape)
print("Target shape: ", iris_dataset_pandas['classe'].shape)
print("Target details ", pd.Categorical(iris_dataset_pandas['classe']).categories)
print("Feature names: ", iris_dataset_pandas.columns)
# sprint("Feature names: ", iris_dataset['feature_names'])

Data shape: (150, 5)
Target shape: (150,)
Target details Index([0.0, 1.0, 2.0], dtype='float64')
Feature names: Index([sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
                    'petal width (cm)', 'classe'],
                    dtype='object')

In [4]: print("Description of dataset: ", iris_dataset['DESCR'][:1300])

Description of dataset: .. _iris_dataset:

Iris plants dataset
-----

**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
- Iris-Setosa
- Iris-Versicolour
- Iris-Virginica

:Summary Statistics:

=====
      Min      Max      Mean      SD      Class Correlation
=====
sepal length:  4.3   7.9   5.84   0.83   0.7826
sepal width:   2.0   4.4   3.85   0.43   -0.4194
petal length:  1.0  6.9   3.76   1.76   0.9490 (high!)
petal width:   0.1   2.5   1.20   0.76   0.9565 (high!)
=====

:Missing Attribute Values: None
:class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL@PLUio.arc.nasa.gov)
:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong dat

In [5]: # Visualisation des corrélations d'attributs
import seaborn as sns
sns.pairplot(iris_dataset_pandas, hue='classe', markers=["o", "s", "D"], palette='deep')

Out[5]: <seaborn.axisgrid.PairGrid at 0x107e82190>
```

Description des attributs:

- Corrélation forte entre certaines variables :
 - corrélation élevée entre la longueur du pétale (petal length) et la largeur du pétale (petal width).
- Faible corrélation entre les sépales et les pétales
- Présence d'indépendance des attributs.

Question 2 : Prétraitement des données

- Le jeu de données nécessite-t-il un prétraitement (tel que le nettoyage, la transformation, etc.) avant l'application des techniques de forage de données ?
- Les données cibles sont-elles présentes ? Si oui, de quel type d'apprentissage s'agit-il (apprentissage supervisé ou non supervisé) ?
- Ainsi, opteriez-vous pour des techniques de classification ou des techniques de régression ? Justifiez votre choix et vos analyses.

Corrigé Question 2

- Le jeu de données nécessite-t-il un prétraitement (tel que le nettoyage, la transformation, etc.) avant l'application des techniques de forage de données ?

Non, car il n'y a aucune donnée manquante ni bruitée. De plus, la présence de données aberrantes aurait été identifiée lors de notre précédente visualisation (la corrélation des valeurs des attributs).
- Les données cibles sont-elles présentes ? Si oui, de quel type d'apprentissage s'agit-il (apprentissage supervisé ou non supervisé) ?

Oui, les différentes espèces d'Iris (présentes dans la colonne 'classe') sont bien présentes. Il s'agit d'un apprentissage supervisé, car toutes nos données d'observation appartiennent à une classe spécifique (Setosa, Versicolor ou Virginica).
- Ainsi, opteriez-vous pour des techniques de classification ou des techniques de régression ? Justifiez votre choix et vos analyses.

Nous opterons pour des techniques de classification, car nos données sont de nature catégorique (0 : Setosa, 1 : Versicolor, et 2 : Virginica).

Question 3 : Clustering

- 3-1)
- Appliquez une décomposition en deux composantes principales (2-CP) et trois composantes principales (3-CP) sur le jeu de données. Affichez une visualisation des données après transformation.
- 3-2)
- Appliquez la méthode t-SNE sur le jeu de données. Affichez une visualisation des données après transformation.
- 3-3)
- Que pouvez-vous conclure concernant les deux méthodes utilisées, à savoir PCA et t-SNE ?
- 3-4)
- Instanciez les trois modèles suivants du module `sklearn.cluster`, entraînez-les sur votre jeu de données, puis effectuez des prédictions. Évaluez ensuite les résultats obtenus.
- KMeans
 - AgglomerativeClustering
 - DBSCAN

Corrigé Question 3

- 3-1) Appliquez une décomposition en deux composantes principales (2-CP) et trois composantes principales (3-CP) sur le jeu de données. Affichez une visualisation des données après transformation.
- ```
In [6]: # Utilisation du dataframe
X représente uniquement iris_dataset['data']. Les observations sans les données cibles (les classes ou espèces d'Iris).
X = iris_dataset_pandas.iloc[:, :-1]
y = iris_dataset_pandas.iloc[:, -1]

In [7]: # Importation de la classe PCA
from sklearn.decomposition import PCA

In [8]: # Décomposition 2CP
pca2 = PCA(n_components=2)
pca2.fit(X)
X_2pca = pca2.transform(X)

Afficher format après transformation
print(X_2pca.shape)
(150, 2)

In [9]: # Décomposition 3CP
pca3 = PCA(n_components=3)
pca3.fit(X)
X_3pca = pca3.transform(X)

Afficher format après transformation
print(X_3pca.shape)
(150, 3)

3-2) Appliquez la méthode t-SNE sur votre jeu de données. Affichez une visualisation des données après transformation.
```
- ```
In [10]: # Importation de la classe TSNE
from sklearn.manifold import TSNE

tsne = TSNE()
X_tsne = tsne.fit_transform(X)

# Afficher format après transformation
print(X_tsne.shape)
(150, 2)

• Affichez une visualisation des données après transformation.
```

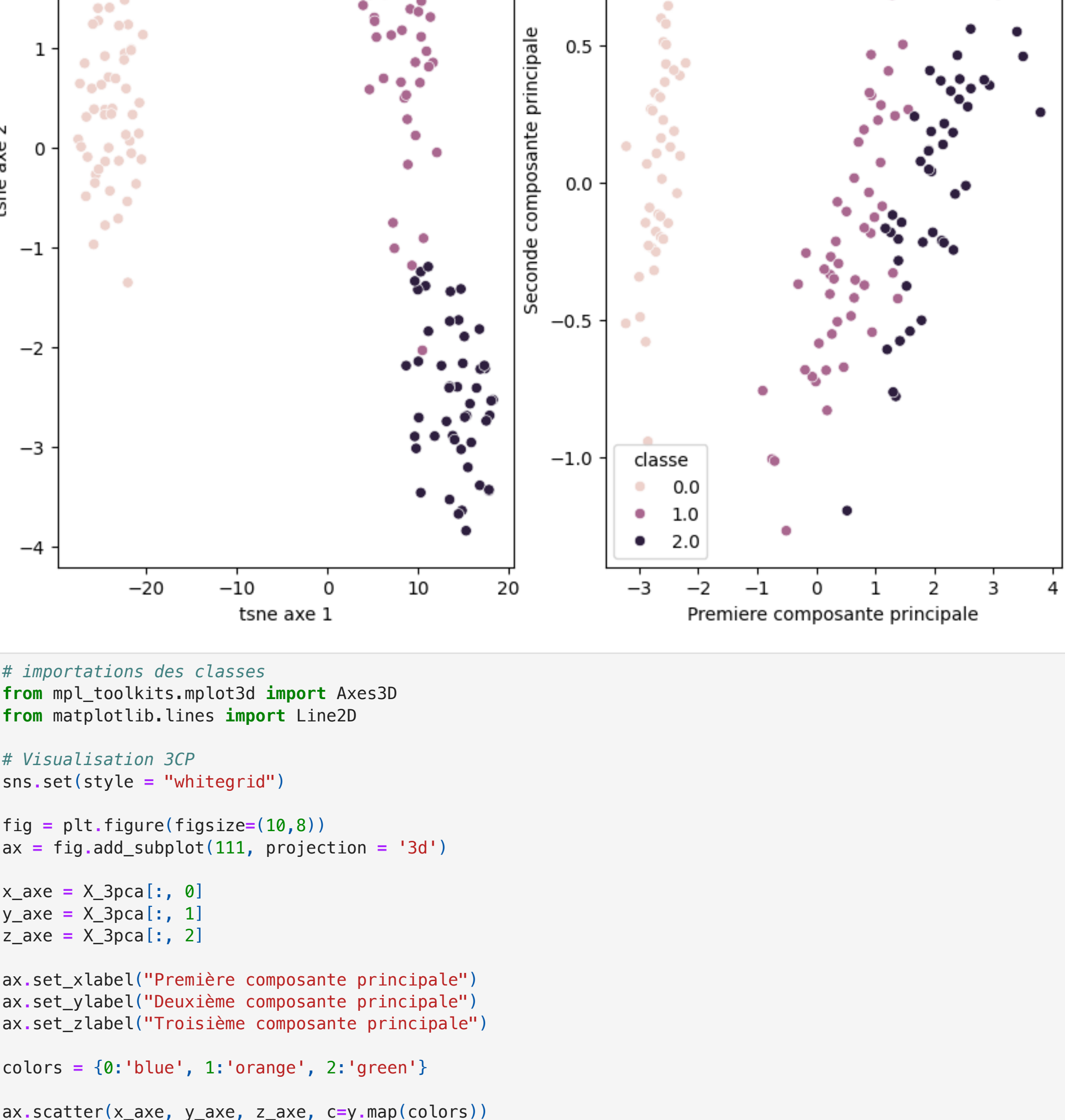
```
In [11]: # Importation de matplotlib.pyplot
import matplotlib.pyplot as plt
%matplotlib inline

In [12]: # Visualisation 2CP et t-SNE
fig, axes = plt.subplots(1, 2, figsize=(10,8))

sns.scatterplot(x=X_2pca[:,0], y=X_2pca[:,1], hue=y, ax=axes[1])
axes[1].set_title('Iris data set')
axes[1].set_xlabel('Première composante principale')
axes[1].set_ylabel('Seconde composante principale')

sns.scatterplot(x=X_tsne[:,0], y=X_tsne[:,1], hue=y, ax=axes[0])
axes[0].set_title('Iris data set')
axes[0].set_xlabel('tsne axe 1')
axes[0].set_ylabel('tsne axe 2')
```

Out[12]: Text(0, 0.5, 'tsne axe 2')



```
In [13]: # Importations des classes
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.lines import Line2D

# Visualisation 3CP
sns.set(style = "whitegrid")
ax = fig.add_subplot(111, projection = '3d')

X_raw = X_3pca[:, 0]
y_axe = X_3pca[:, 1]
Z_axe = X_3pca[:, 2]

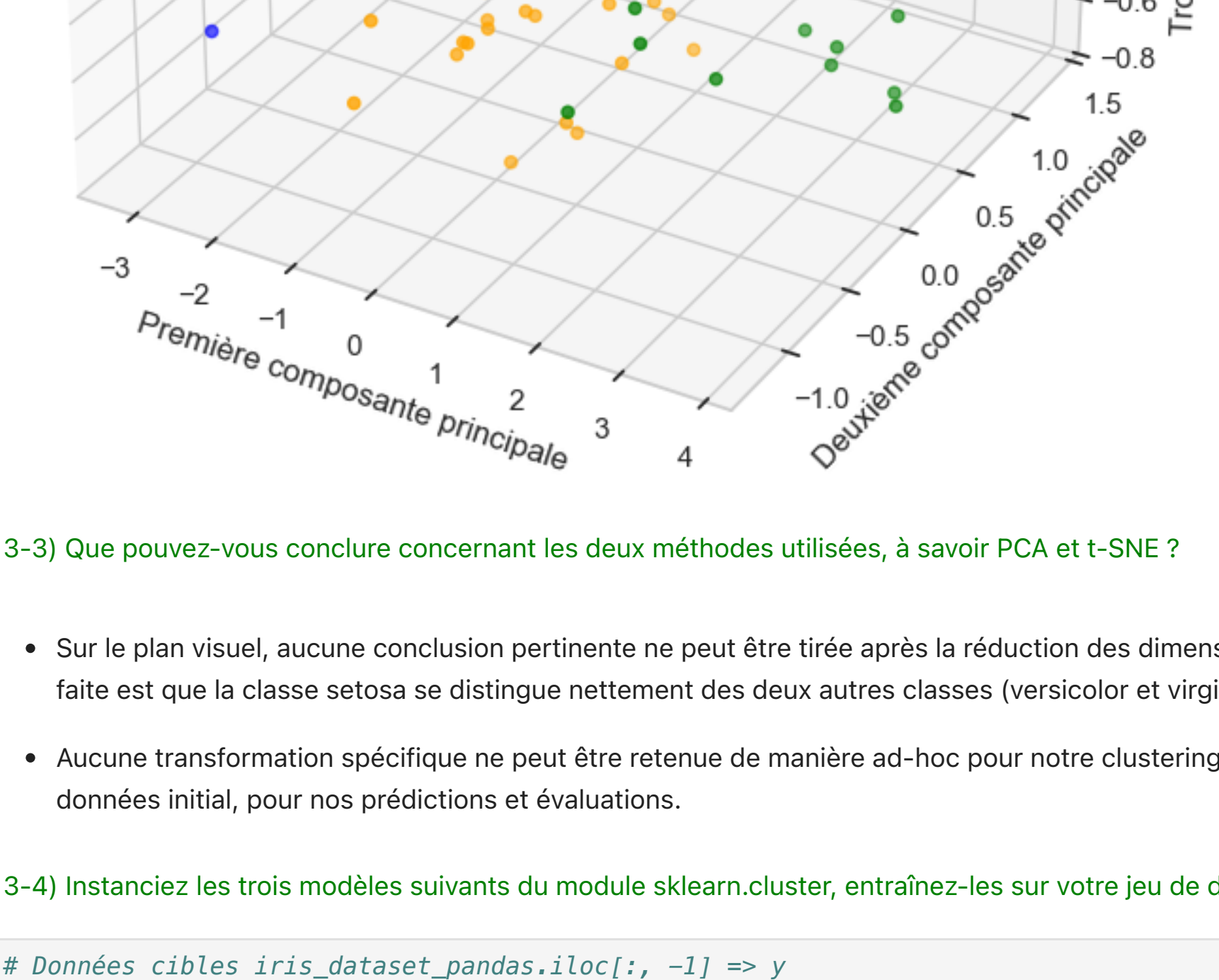
ax.set_xlabel("Première composante principale")
ax.set_ylabel("Deuxième composante principale")
ax.set_zlabel("Troisième composante principale")

colors = {0:'blue', 1:'orange', 2:'green'}

ax.scatter(X_axe, y_axe, Z_axe, c=y, map(colors))

# Ajout de la légende
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=k, markersize=8) for k, v in colors.items()]
ax.legend(title='Classes', handles=handles, bbox_to_anchor=(1.05, 1), loc='upper left')
```

Out[13]: <matplotlib.legend.Legend at 0x133d2b9d0>



- 3-3) Que pouvez-vous conclure concernant les deux méthodes utilisées, à savoir PCA et t-SNE ?

- Sur le plan visuel, aucune conclusion pertinente ne peut être tirée après la réduction des dimensions de notre jeu de données (à l'aide de 2CP, 3CP ou t-SNE). La seule observation qui peut être faite est que la classe setosa se distingue nettement des deux autres classes (versicolor et virginica).
- Aucune transformation spécifique ne peut être retenue de manière ad-hoc pour notre clustering. Nous avons donc choisi de conserver les trois jeux de données transformés, ainsi que le jeu de données initial, pour nos prédictions et évaluations.

- 3-4) Instanciez les trois modèles suivants du module `sklearn.cluster`, entraînez-les sur votre jeu de données, puis effectuez des prédictions. Évaluez ensuite les résultats obtenus.

```
In [14]: # Données cibles iris_dataset_pandas.iloc[:, -1] => y
# Préparation de notre jeu de données brut
X_raw = X_iris_dataset_pandas.iloc[:, :-1]

# Préparation de notre jeu de données 2CP => X_2pca
# Préparation de notre jeu de données 3CP => X_3pca
# Préparation de notre jeu de données t-SNE => X_tsne

In [15]: # Importation des classes
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN

# Instancier les modèles
kmeans = KMeans(n_clusters = 3, n_init='auto')
agclus = AgglomerativeClustering(n_clusters = 3)
dbscan = DBSCAN()

# fit-predict sur nos jeux de données
collection_data = [X_raw, X_2pca, X_3pca, X_tsne]
models = [kmeans, agclus, dbscan]

collection_y_predicted = []
for dataset in collection_data:
    dataset_y_predicted = [model.fit_predict(dataset) for model in models]
    collection_y_predicted.append(dataset_y_predicted)

In [16]: # Evaluation du clustering à partir du clustering réel

# Importation des fonctions
from sklearn.metrics.cluster import adjusted_rand_score

# Création d'un dataframe pour stocker tous les scores et mieux les comparer
df_scores = pd.DataFrame(columns=['data', 'KMeans', 'AgglomerativeClustering', 'DBSCAN'])

# Calcul du score
for dataset_y_predicted in enumerate(collection_y_predicted):
    key_collection = dataset_y_predicted[0]
    model_y_predicted = dataset_y_predicted[1]

# Mapping du dataset
dataset = 'X_raw' if key_collection == 0 else 'X_2pca' if key_collection == 1 else 'X_3pca' if key_collection == 2 else 'X_tsne'

# Ajout du jeu de données dans le dataframe
df_scores.loc[key_collection, 'data'] = dataset

for y_predicted in enumerate(model_y_predicted):
    key_model = y_predicted[0]
    y_m = y_predicted[1]

# Mapping du model
model = 'KMeans' if key_model == 0 else 'AgglomerativeClustering' if key_model == 1 else 'DBSCAN'

# Calcul du score
score = adjusted_rand_score(y, y_m)

# Ajout du score dans le dataframe
df_scores.loc[key_collection, model] = score

# Afficher le dataframe
print(df_scores)
```

	data	KMeans	AgglomerativeClustering	DBSCAN
0	X_raw	0.780238	0.731199	0.520619
1	X_2pca	0.716342	0.744526	0.551005
2	X_3pca	0.716342	0.732298	0.531108
3	X_tsne	0.817753	0.759199	0.817567

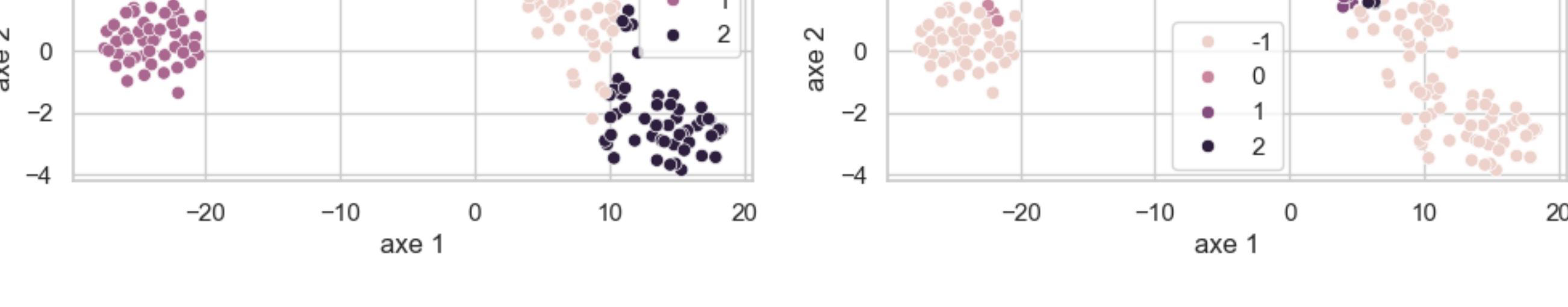
Observations

- Les algorithmes KMeans et AgglomerativeClustering donnent des résultats meilleurs que DBSCAN
- DBSCAN a tendance à confondre les deux classes versicolor et virginica, qui ne se distinguent pas clairement en termes de densité (cf. illustration ci-dessous).
- Il est possible d'utiliser différentes métriques pour évaluer la qualité de nos clusters, permettant ainsi de déterminer le meilleur choix entre KMeans et AgglomerativeClustering.

```
In [17]: # Visualisation DBSCAN vs KMeans
y_Xtsne_kmeans = collection_y_predicted[3][0]
y_Xtsne_dbscan = collection_y_predicted[3][2]

fig, axes = plt.subplots(1, 2, figsize=(12,2))

i=0
for y_m in [y_Xtsne_kmeans, y_Xtsne_dbscan]:
    sns.scatterplot(x=X_tsne[:,0], y=X_tsne[:,1], hue=y_m, ax=axes[i])
    model_name = 'KMeans' if i==0 else 'DBSCAN'
    axes[i].set_title(model_name)
    axes[i].set_xlabel('axe 1')
    axes[i].set_ylabel('axe 2')
    i += 1
```



In []:

In []:

In []: