Visualisation des données en 2D ou 3D

Dans ce notebook nous nous intéressons à la visualisation des données en 2D ou 3D. Il s'intéresse à la projection des données dans un espace 2D (resp. 3D). Cela peut être très utile avant de faire de la classification pour avoir une idée de la répartition des classes, d'outlier, etc.

Installation

Avant de commencer, il est nécessaire de déjà posséder dans son environnement toutes les librairies utiles. Dans la seconde cellule nous importons toutes les librairies qui seront utiles à ce notebook. Il se peut que, lorsque vous lanciez l'éxecution de cette cellule, une soit absente. Dans ce cas il est nécessaire de l'installer. Pour cela dans la cellule suivante utiliser la commande :

! pip install nom_librairie

Attention : il est fortement conseillé lorsque l'une des librairies doit être installer de relancer le kernel de votre notebook.

Remarque : même si toutes les librairies sont importées dès le début, les librairies utiles pour des fonctions présentées au cours de ce notebook sont ré-importées de manière à indiquer d'où elles viennent et ainsi faciliter la réutilisation de la fonction dans un autre projet.

Dans ce notebook nous utilisons UMAP pour faire de la réduction de dimensions. Les librairies suivantes doivent forcément être installées

```
# utiliser cette cellule pour installer les librairies manquantes
# pour cela il suffit de taper dans cette cellule : !pip install nom_librairie_m
# d'exécuter la cellule et de relancer la cellule suivante pour voir si tout se
# recommencer tant que toutes les librairies ne sont pas installées ...

# sous Colab il faut déjà intégrer ces deux librairies

!pip install umap-learn[plot]
!pip install holoviews
!pip install -U ipykernel

# eventuellement ne pas oublier de relancer le kernel du notebook
```

```
Requirement already satisfied: panel>=0.8.0 in /usr/local/lib/python3.8/dis
Requirement already satisfied: pyviz-comms>=0.7.4 in /usr/local/lib/python3
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dis
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.8/d
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.8/di
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8
Requirement already satisfied: bleach in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: markdown in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: fsspec>=0.6.0 in /usr/local/lib/python3.8/di
Requirement already satisfied: partd>=0.3.10 in /usr/local/lib/python3.8/di
Requirement already satisfied: cloudpickle>=1.1.1 in /usr/local/lib/python3
Requirement already satisfied: multipledispatch>=0.4.7 in /usr/local/lib/py
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-p
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dis
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/pyth
Requirement already satisfied: locket in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: webencodings in /usr/local/lib/python3.8/dis
Building wheels for collected packages: pynndescent, umap-learn, datashape
  Building wheel for pynndescent (setup.py) ... done
  Created wheel for pynndescent: filename=pynndescent-0.5.8-py3-none-any.wh
  Stored in directory: /root/.cache/pip/wheels/1c/63/3a/29954bca1a27ba100ed
  Building wheel for umap-learn (setup.py) ... done
  Created wheel for umap-learn: filename=umap_learn-0.5.3-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/a9/3a/67/06a8950e053725912e6
  Building wheel for datashape (setup.py) ... done
  Created wheel for datashape: filename=datashape-0.5.2-py3-none-any.whl si
  Stored in directory: /root/.cache/pip/wheels/6d/79/c4/c425774559165f472d3
Successfully built pynndescent umap-learn datashape
Installing collected packages: datashape, pynndescent, umap-learn, datashad
Successfully installed datashader-0.14.4 datashape-0.5.2 pynndescent-0.5.8
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/cola
Requirement already satisfied: holoviews in /usr/local/lib/python3.8/dist-p
Requirement already satisfied: panel>=0.8.0 in /usr/local/lib/python3.8/dis
Requirement already satisfied: numpy>=1.0 in /usr/local/lib/python3.8/dist-
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.8/d
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-p
Requirement already satisfied: colorcet in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: param<2.0,>=1.9.3 in /usr/local/lib/python3.
Requirement already satisfied: pyviz-comms>=0.7.4 in /usr/local/lib/python3
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dis
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pyt
Requirement already satisfied: bokeh<2.4.0,>=2.3.0 in /usr/local/lib/python
Requirement already satisfied: bleach in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: markdown in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: pyct>=0.4.4 in /usr/local/lib/python3.8/dist
```

```
Requirement already satisfied: pyct>=0.4.4 in /usr/local/lib/python3.8/dist
Requirement already satisfied: tqdm>=4.48.0 in /usr/local/lib/python3.8/dis
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.8/di
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/p
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.8/dis
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.8/dis
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.8/dist
```

```
# Importation des différentes librairies utiles pour le notebook

#Sickit learn met régulièrement à jour des versions et
#indique des futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# librairies générales
import pickle
import pandas as pd
from scipy.stats import randint
import numpy as np
import string
import time
import base64
import re
import sys
import copy


# librairie affichage
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
import plotly.graph_objs as go
import plotly.offline as py
import plotly.express as px

from sklearn.metrics import confusion_matrix
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# librairies NLTK
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk import word_tokenize


nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
stop_words = set(stopwords.words('english'))

# Umap
import umap.plot
from umap import UMAP
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Pour pouvoir sauvegarder sur votre répertoire Google Drive, il est nécessaire de fournir une autorisation. Pour cela il suffit d'éxecuter la ligne suivante et de saisir le code donné par Google.

```
# pour monter son drive Google Drive local
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Corriger éventuellement la ligne ci-dessous pour mettre le chemin vers un répertoire spécifique dans votre répertoire Google Drive :

```
import sys
my_local_drive='/content/gdrive/My Drive/Colab Notebooks/ML_FDS'
# Ajout du path pour les librairies, fonctions et données
sys.path.append(my_local_drive)
# Se positionner sur le répertoire associé
%cd $my_local_drive

%pwd
```

```
/content/gdrive/My Drive/Colab Notebooks/ML_FDS
'/content/gdrive/My Drive/Colab Notebooks/ML_FDS'
```

```
# fonctions utilities (affichage, confusion, etc.)
from MyNLPUtilities import *
```

Visualisation des données

Il est intéressant lorsque l'on a un jeu de données de regarder aussi comment ces dernières se répartissent dans l'espace en 2D ou en 3D. Cela peut permettre de déterminer des outliers, de voir des regroupements qui indiquent que souvent ces données sont plus facilement prédictibles ou au contraire des points de différentes classes très proches qui seront difficilement prédictibles, etc.

Etant donné que les données ont un nombre trop grand de dimensions pour être visualisées, une solution consiste à utiliser de la réduction de dimensions.

Il existe différentes méthodes comme par exemple :

- PCA (*Principal Component Analysis - Analyse en Composantes Principales*) [Hotelling1933] qui consiste à transformer des variables corrélées entre elles (dites « corrélées » en statistique) en nouvelles variables décorrélées les unes des autres, i.e. les composantes principales. Ces nouvelles variables sont nommées « composantes principales ».
- T-SNE (*t-distributed Stochastic Neighbor Embedding*) [VanderMaaten2008] , plus récente, 2008, et très utilisée en visualisation, utilise une interprétation probabiliste des proximités, i.e. distribution de probabilité, au lieu des matrices de variance et co-variance de PCA.
- UMAP (*Uniform Manifold Approximation and Projection*) [McInnes et al. 2018], John Healy, James Melville, créé en 2018, utilise une technique de réduction de dimension non linéaire.

[Hotelling1933] H Hotelling. « Analysis of a Complex of Statistical Variables with Principal Components », 1933, Journal of Educational Psychology.

[VanderMaaten2008] L.J.P. van der Maaten and G.E. Hinton. « Visualizing High-Dimensional Data Using t-SNE », Journal of Machine Learning Research, vol. 9, novembre 2008, p. 2579–2605.

[McInnes et al. 2018] Leland McInnes, John Healy and James Melville. « UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction », arXiv:1802.03426, 2018.

Visualisation des données IRIS

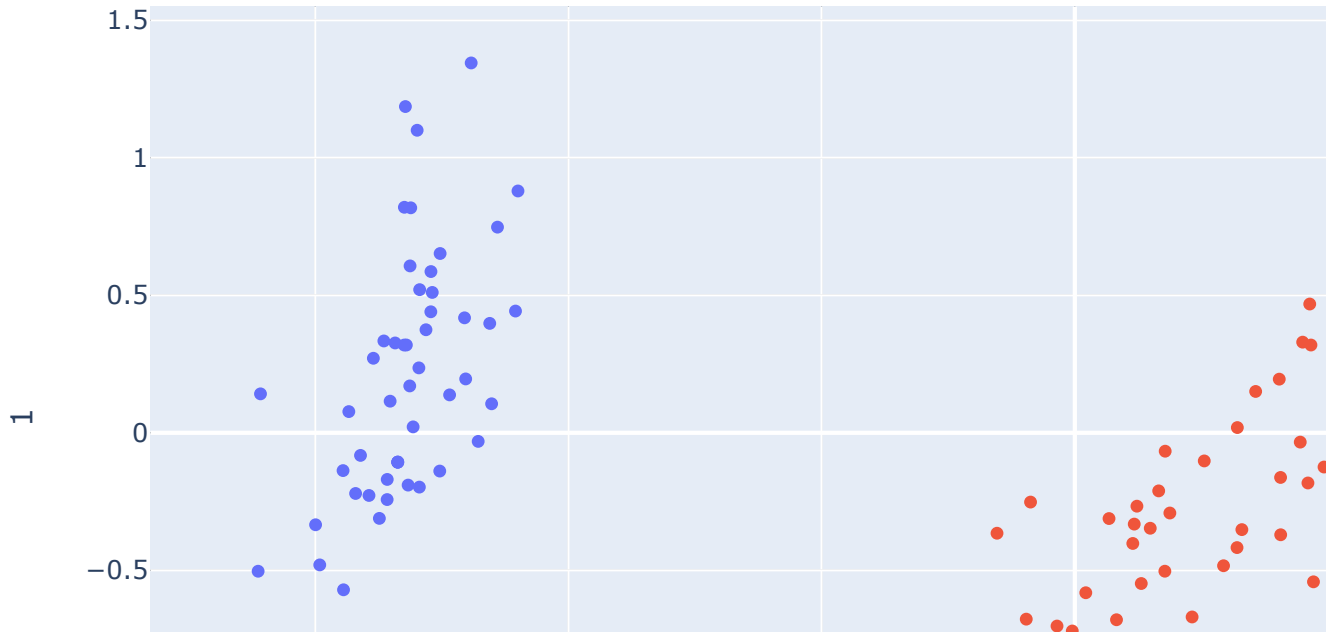
Dans un premier temps, nous illustrons la visualisation à l'aide du jeu de données IRIS.

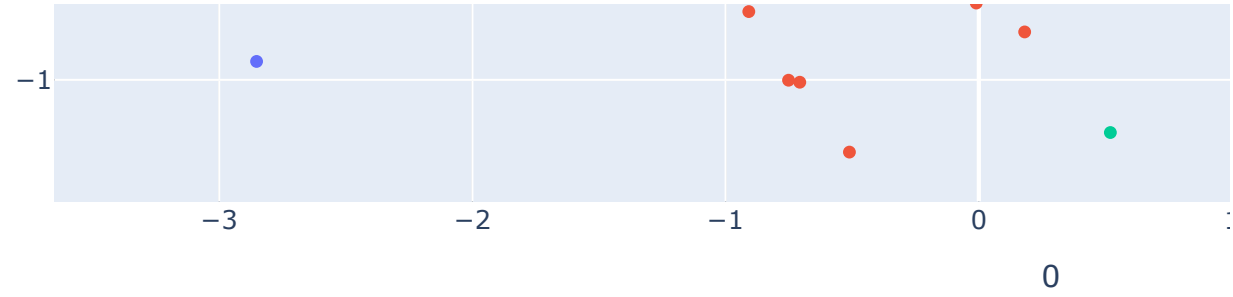
La librairie plotly offre la possibilité de facilement faire de l'ACP, ou d'utiliser T-SNE et UMAP.

```
df = px.data.iris()
X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
```

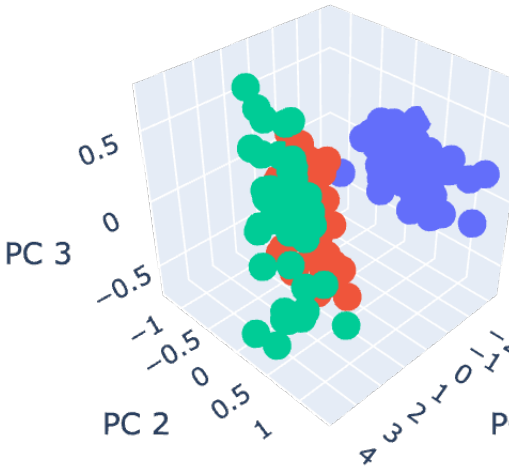
ACP pour IRIS

```
# 2D
pca = PCA(n_components=2, random_state=0)
components = pca.fit_transform(X)

fig = px.scatter(components, x=0, y=1, color=df['species'])
fig.show()

#3D
pca = PCA(n_components=3, random_state=0)
components = pca.fit_transform(X)

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=df['species'],
    title='ACP',
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}
)
fig.show()
```





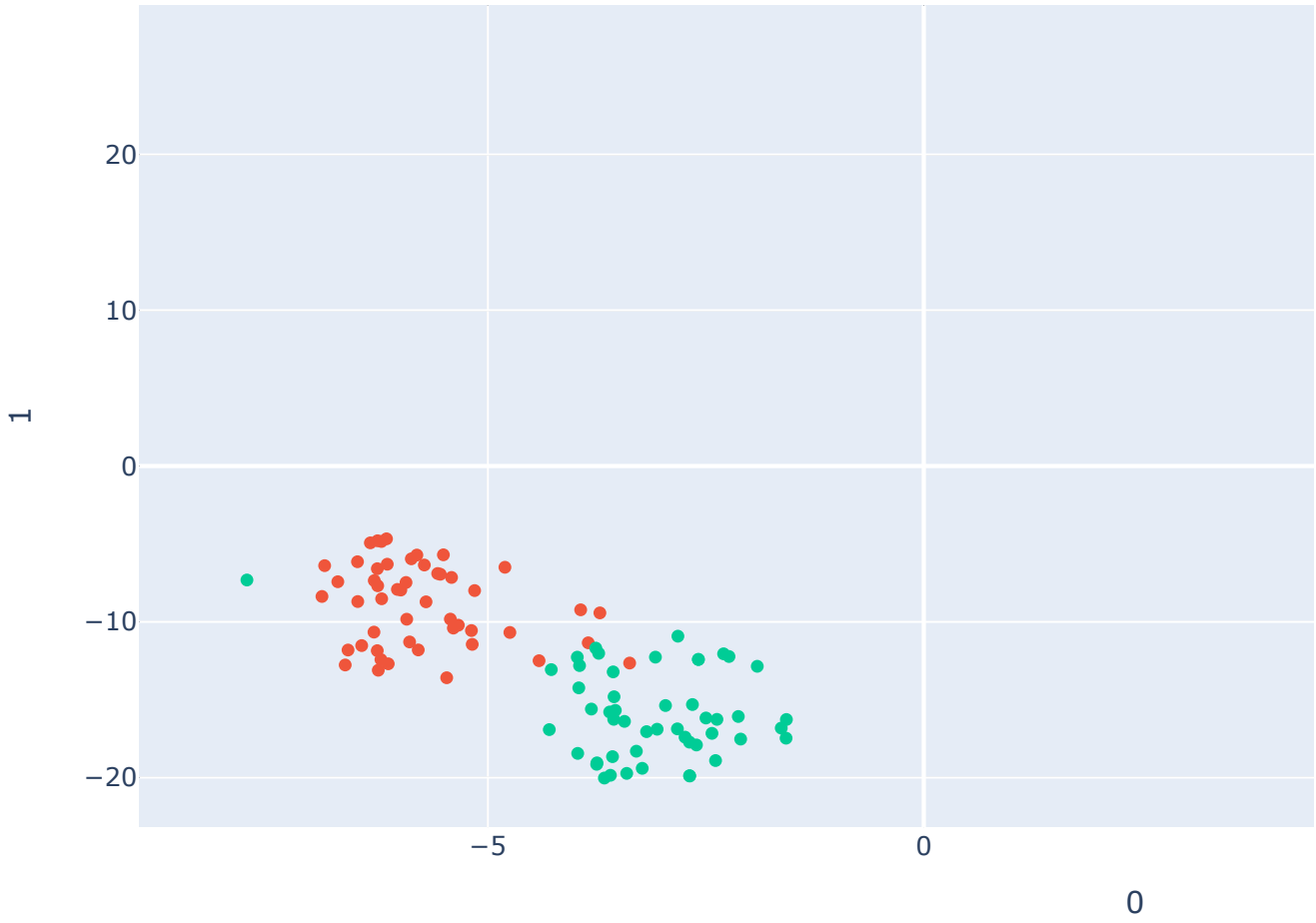
ACP

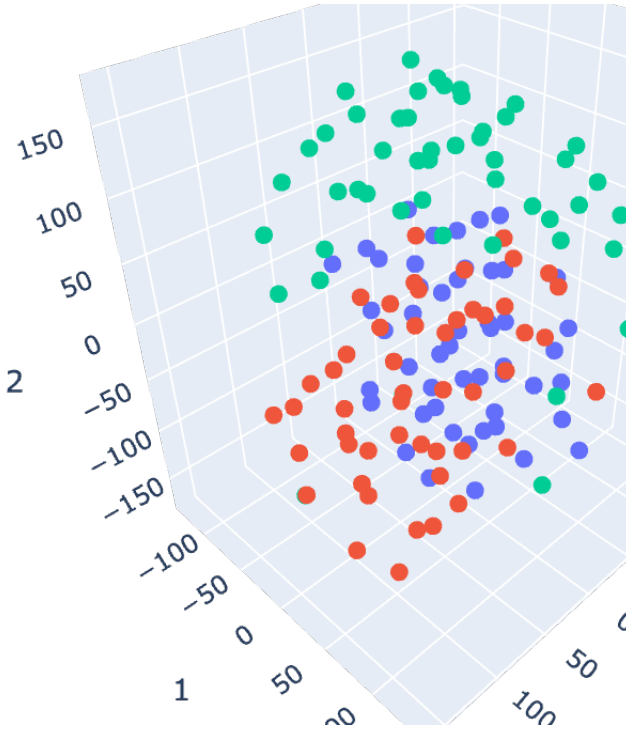


t-SNE pour IRIS

```
# 2D
tsne = TSNE(n_components=2, random_state=0)
projections = tsne.fit_transform(X)

fig = px.scatter(
    projections, x=0, y=1,
```

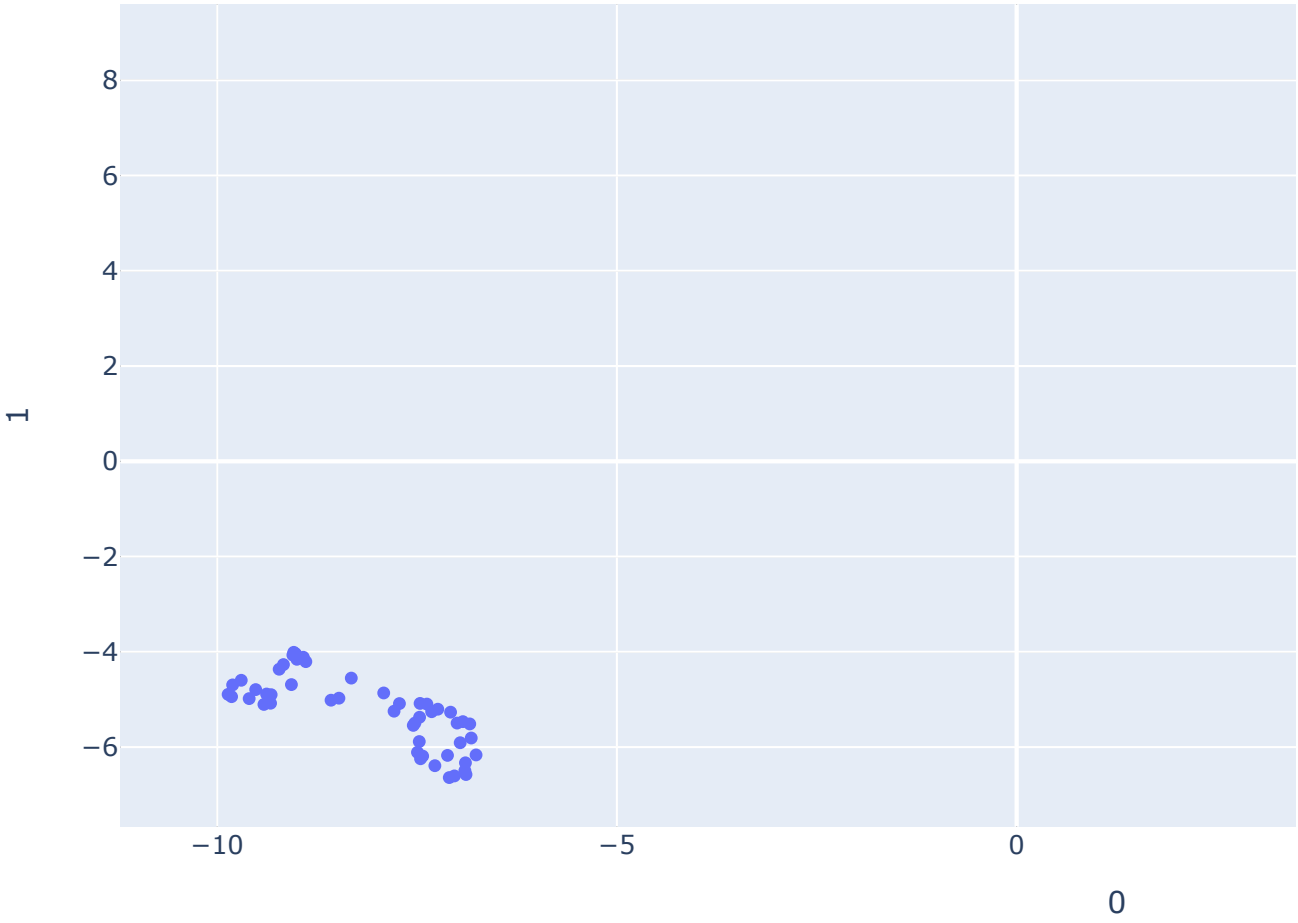
```
    color=df.species, labels={'color': 'species'}
)
fig.show()

# 3D
tsne = TSNE(n_components=3, random_state=0)
projections = tsne.fit_transform(X)

fig = px.scatter_3d(
    projections, x=0, y=1, z=2,
    color=df.species, labels={'color': 'species'}
)
fig.update_traces(marker_size=5)
fig.show()
```

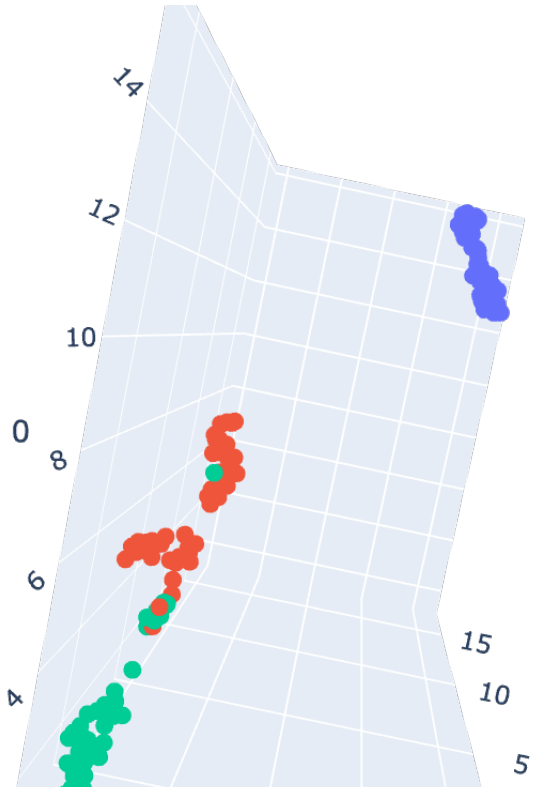




UMAP pour IRIS

```
# 2D
umap = UMAP(n_components=2, init='random', random_state=0)
projection = umap.fit_transform(X)

fig = px.scatter(
    projection, x=0, y=1,
    color=df.species, labels={'color': 'species'}
)
fig.show()

# 3D
umap = UMAP(n_components=3, init='random', random_state=0)
projection = umap.fit_transform(X)
fig = px.scatter_3d(
    projection, x=0, y=1, z=2,
    color=df.species, labels={'color': 'species'}
)
fig.update_traces(marker_size=5)
fig.show()
```


8
6
4
2
0
−2
−4
−6
1
−10
−5
0
0


14
12
10
0
8
6
4
15
10
5

Visualisation de données textuelles

Il est également possible de visualiser comment des données textuellees se positionnent dans l'espace. Il suffit de transformer le document en vecteur et de réduire les dimensions pour les visualiser.

```
!wget https://www.lirmm.fr/~poncelet/Ressources/ReviewsLabelled.csv
```

```
df = pd.read_csv("ReviewsLabelled.csv", names=['sentence','sentiment','source'],


# selection des données
X=df.sentence

tfidf=TfidfVectorizer()
vector_tfidf=tfidf.fit_transform(X)
```

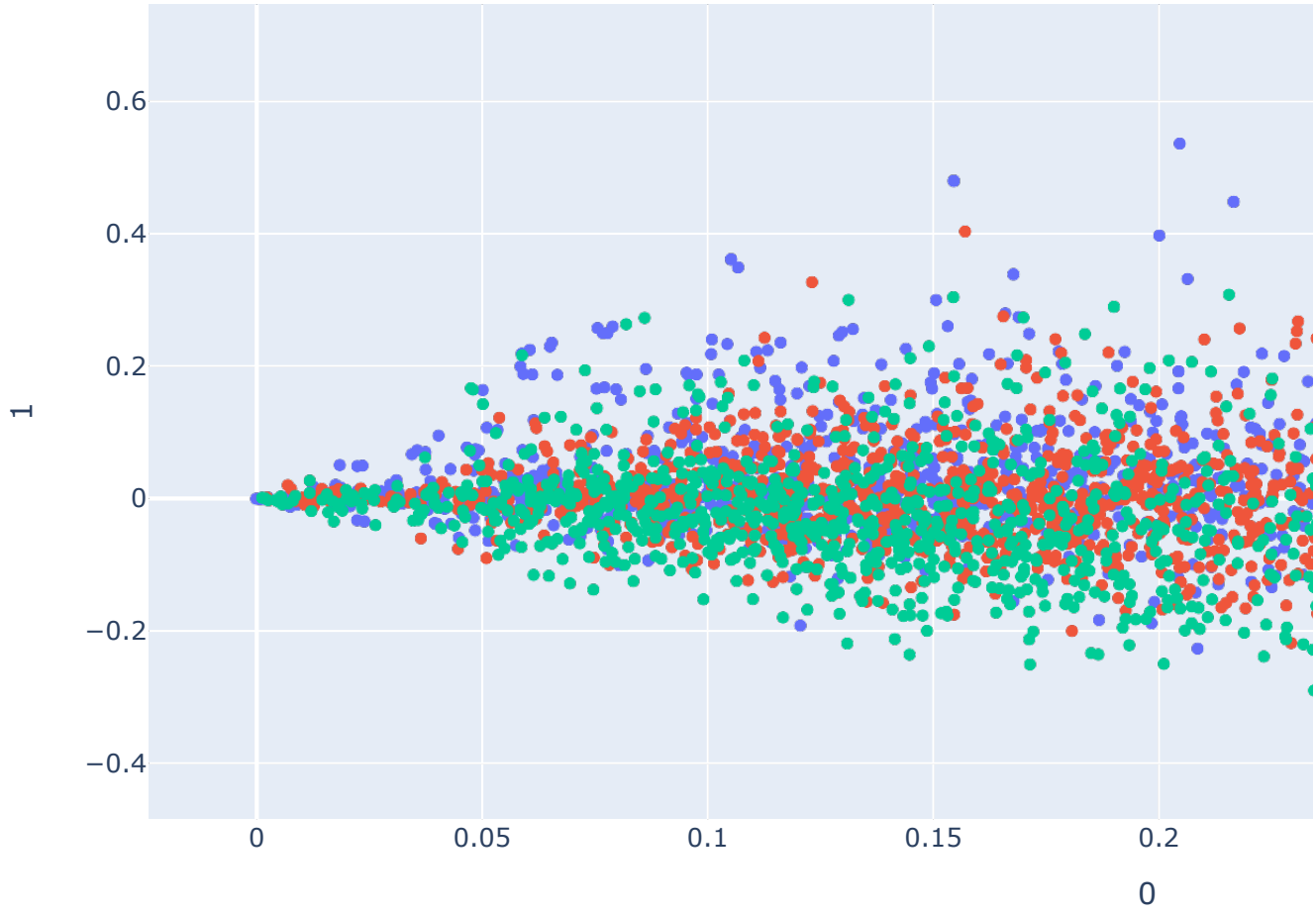
Sur ce jeu de données, il est possible de regarder l'opinion (la classe positive et négative) ou bien les sources. Dans la suite, nous regardons les sources. Pour afficher par rapport à l'opinion, il suffit de décommenter la ligne correspondante dans la cellule suivante et de remplacer df['source'] dans les px.scatter par df['sentiment']

```
y=df.source
#y=df.sentiment
```

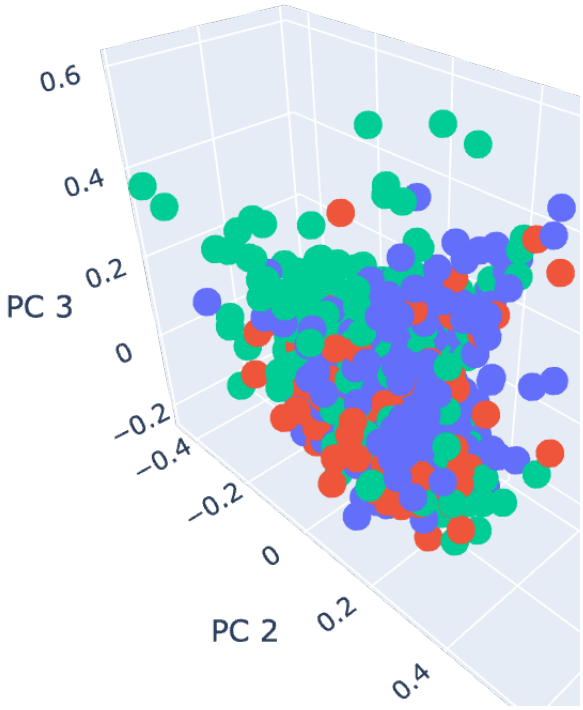
ACP pour les textes

Attention l'ACP ne fonctionne pas bien sur des matrices sparses. Ce qui est notamment le cas lors de la transformation avec tf_idf. Il faut dans ce cas utiliser la librairie TruncatedSVD qui permet de faire de la réduction de dimension.

```
from sklearn.decomposition import TruncatedSVD

#2D
svd=TruncatedSVD(n_components=2, random_state=0)
components = svd.fit_transform(vector_tfidf)

fig = px.scatter(components, x=0, y=1, color=df['source'])
fig.show()


#3D
svd=TruncatedSVD(n_components=3, random_state=0)
components = svd.fit_transform(vector_tfidf)

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=df['source'],
    title='TruncatedSVD',
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}
)
fig.show()
```



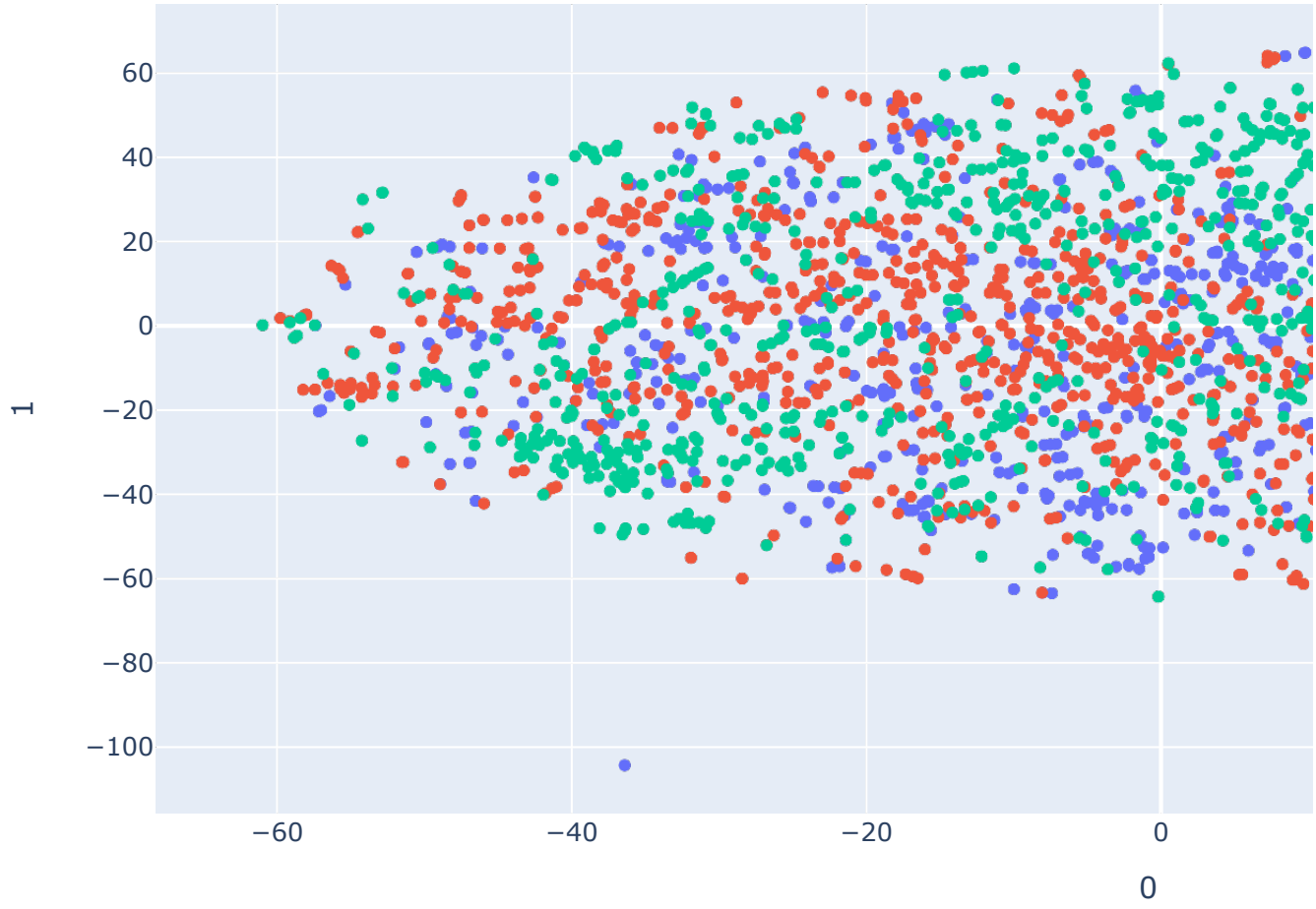
TruncatedSVD

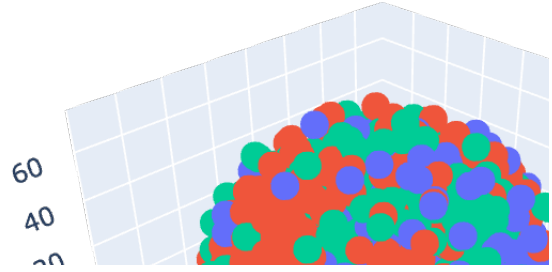


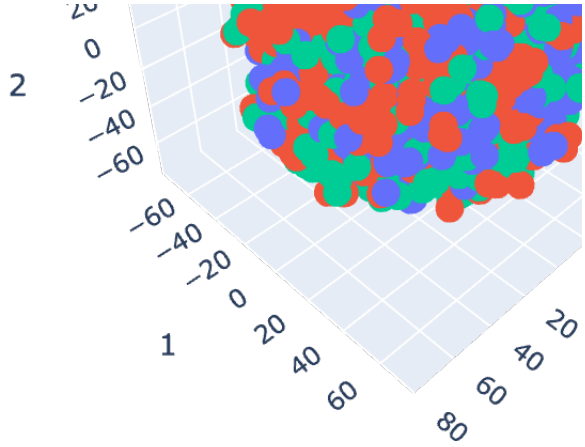
t-SNE pour texte

```
# 2D
tsne = TSNE(n_components=2, random_state=0)
projections = tsne.fit_transform(vector_tfidf)

fig = px.scatter(
    projections, x=0, y=1,
    color=df.source, labels={'color': 'sources'}
)
fig.show()


# 3D
tsne = TSNE(n_components=3, random_state=0)
projections = tsne.fit_transform(vector_tfidf)
```

```
fig = px.scatter_3d(
    projections, x=0, y=1, z=2,
    color=df.source, labels={'color': 'sources'}
)
fig.update_traces(marker_size=5)
fig.show()
```

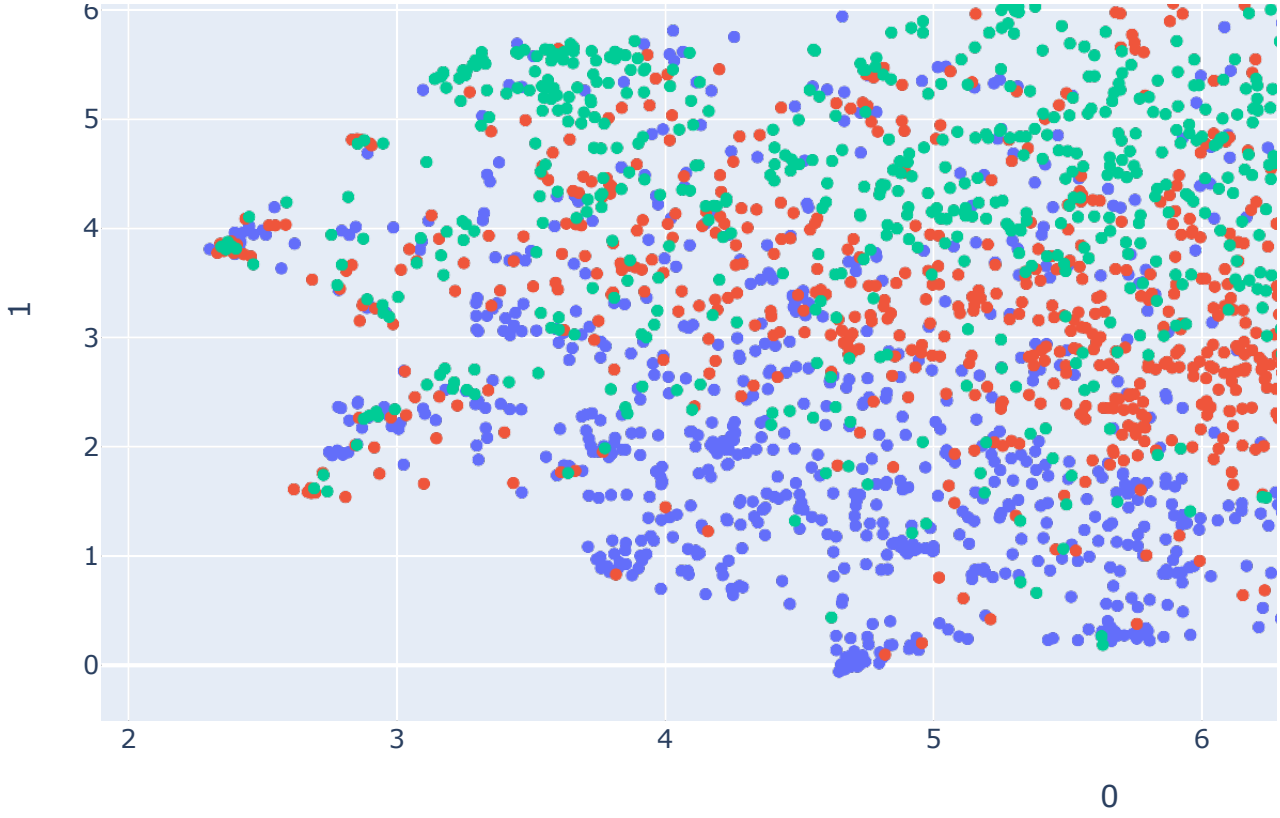


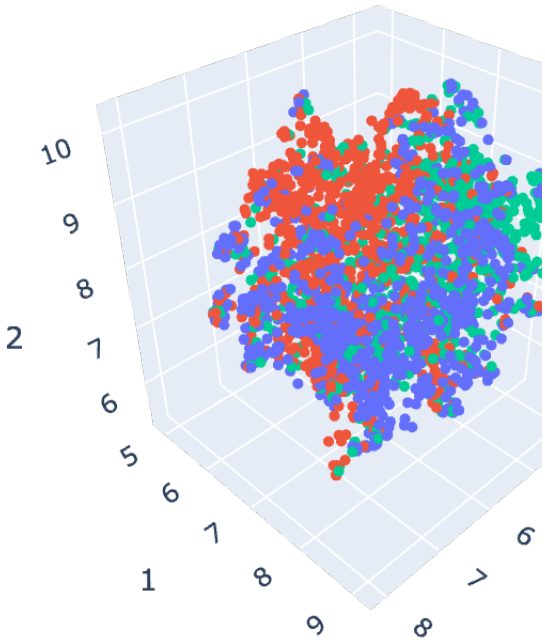




UMAP pour texte

```
# 2D
umap = UMAP(n_components=2, init='random', random_state=0)
projection = umap.fit_transform(vector_tfidf)

fig = px.scatter(
    projection, x=0, y=1,
    color=df.source, labels={'color': 'sources'}
)

fig.show()

# 3D
umap = UMAP(n_components=3, init='random', random_state=0)
projection = umap.fit_transform(vector_tfidf)

fig = px.scatter_3d(
    projection, x=0, y=1,z=2,
    color=df.source, labels={'color': 'sources'}
)
fig.update_traces(marker_size=3)
fig.show()
```


6
5
4
3
2
1
0
1
2
3
4
5
6
0


10
9
8
7
6
5
2
6
7
8
9
1
8
7
6

Produits payants Colab - Résilier les contrats ici

✓ 0 s terminée à 16:03