EPI ECOLE PLURISDISCIPLINAIRE INTERNATIONALE AU : 2022-2023	TP2	IA Marwa Jabberi Slama Yousra Hadj Hassen

Objectifs :

Ce TP est composé de deux parties :

Partie 1 : La découverte des bases de l'**apprentissage supervisé et non supervisé**.

Partie 2 : La pratique avec le langage Python et la librairie **Scikit-Learn** !

Scikit Learn

Il s'agit d'une librairie (package) d'apprentissage automatique couvrant l'ensemble de :

- Les types d'apprentissage : supervisé, non supervisé, par renforcement, par transfert
- Les algorithmes :
  - Linear Regression (régression linéaire),
  - Logistic Regression (régression logistique),
  - Decision Tree (arbre de décision),
  - SVM (machines à vecteur de support),
  - Naive Bayes (classification naïve bayésienne),
  - KNN (Plus proches voisins),
  - Dimensionality Reduction Algorithms,
  - Gradient Boost & Adaboost,
  - Réseaux de neuronnes

- Elle dispose d'une excellente documentation fournissant de nombreux exemples
- Elle dispose d'une API uniforme entre tous les algorithmes, ce qui fait qu'il est facile de basculer de l'un à l'autre
- Elle est très bien intégrée avec les Librairies Pandas et Seaborn
- Elle dispose d'une grande communauté et de plus de 800 contributeurs référencés sur GitHub !

Composition de SkLearn

Les données

Typiquement, des tableaux Numpy ou Pandas ou Python

- Les lignes représentent les enregistrements
- Les colonnes les attributs (hauteur, longueur, couleur, autre information)

- Une donnée est un vecteur de paramètres, généralement des réels, mais les entiers, booléens et valeurs discrètes sont autorisées dans certains cas
- Les labels peuvent être de différents types, généralement des entiers ou chaînes
- Les labels sont contenus dans un tableau à une dimension, sauf rares cas où ils peuvent être dans le vecteur de paramètres

Prédiction

L'algorithme de prédiction est représenté par une classe.

- Il faut commencer par choisir l'algorithme à utiliser (*prédicteur/ classifieur/ estimateur).* Les algorithmes sont des classes Python. Les données sont toujours des tableaux Numpy/Scipy/Pandas/Python
- Il faut préciser les paramètres, appelés ***hyperparamètres*** en instanciant la classe
- Alimentation avec la fonction **fit** dans le cas d'un **apprentissage supervisé**
- Lancement de la prédiction sur un ensemble de valeurs via la fonction **predict** parfois appelée **transform** dans le cas de l'apprentissage non supervisé

Travail demandé :

Idée de base

Etant donné un ensemble de données D, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver **une fonction de mapping entre les variables prédictives en entrée X et la variable à prédire Y**.

Etude de cas

- Classer une fleur selon des critères observables (type de pétales, sépale, type des feuilles, forme des feuilles, ...).
- Demander à l'ordinateur de déterminer automatiquement l'espèce d'une **nouvelle plante** en fonction de la mesure des dimensions de ses sépales et pétales.
- La machine doit être capable de construire **sa décision** à partir de la connaissance extraite des mesures réalisées par un être humain.

Réellement, on va donner à l'ordinateur un jeu de données déjà classé et annoté et lui demander de classifier de nouvelles données à partir de celui-ci en se basant sur la base de connaissance qu'il a construit par l'apprentissage.

C'est un cas d'apprentissage ……

Iris Dataset :

La base de données **IRIS** se compose de trois classes 'Setosa', 'Versicolor' et 'Virginica'.

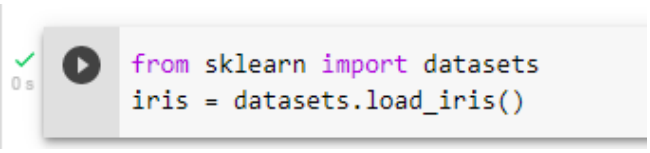
Chaque espèce a quatre propriétés : longueur et largeur de sépales ainsi que longueur et largeur de pétales.

Chaque classe contient 50 échantillons qui signifie 150 observations (50 observations par espèce).

Les étiquettes **Y** peuvent donc appartenir à l'ensemble **{0, 1, 2}**. Il s'agit donc d'une classification ……..

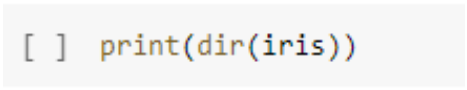
Partie 1

1. Chargement de la base de données contenant un grand nombre d'informations sur les fleus



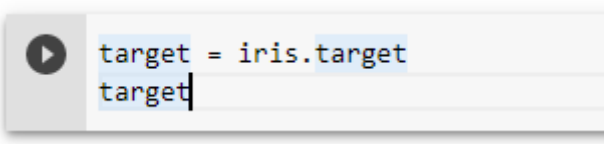
```
from sklearn import datasets
iris = datasets.load_iris()
```

2. Pour découvrir le contenu / les champs de la base, tapez la ligne de code suivante :



```
[ ]  print(dir(iris))
```

3. Afficher les caractéristiques d'une espèce de fleur dans la base de données iris.
4. Affichez les valeurs des 5 premières fleurs de la base de données iris.
5. Affichez la liste des espèces connues (les labels).
6. On veut afficher un tableau indiquant le numéro de l'espèce de chaque enregistrement.



```
target = iris.target
target
```

Interprétation :
……
……
……
……

7. Affichez le nombre d'échantillon pour chaque classe.
8. Tapez ce code permettant la visualisation des données de la base Iris.

```
data=iris.data
```

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
```

```
fig = plt.figure(figsize=(8, 4))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
ax1 = plt.subplot(1,2,1)

clist = ['violet', 'yellow', 'blue']
colors = [clist[c] for c in iris.target]

ax1.scatter(data[:, 0], data[:, 1], c=colors)
plt.xlabel('Longueur du sepal (cm)')
plt.ylabel('Largueur du sepal (cm)')

ax2 = plt.subplot(1,2,2)

ax2.scatter(data[:, 2], data[:, 3], color=colors)

plt.xlabel('Longueur du petal (cm)')
plt.ylabel('Largueur du petal (cm)')
```

```
# Légende
for ind, s in enumerate(iris.target_names):
    # on dessine de faux points, car la légende n'affiche que les points ayant un label
    plt.scatter([], [], label=s, color=clist[ind])

plt.legend(scatterpoints=1, frameon=False, labelspacing=1
           , bbox_to_anchor=(1.8, .5) , loc="center right", title='Espèces')
plt.plot()
```

9. Que remarquez-vous ?

……
……

Apprentissage :

Dans cette partie, nous allons utiliser la classification *Naive Bayes* qui suppose que chaque classe est construite à partir d'une distribution Gaussiènne alignée. Elle n'impose pas de définir d'hyperparamètres et est très rapide.

10. Les étapes suivantes permettent la création du classifieur

```
[22] from sklearn.naive_bayes import GaussianNB
```

```
[23] clf = GaussianNB()
```

11. Démarrage du training : Quelles sont les données nécessaires pour le training ? Expliquez.

……
……
……
……

12. Examinez les paramètres du classifieur, affichez les résultats de la prédiction puis tapez ses lignes de code pour voir le nombre d'erreurs.

```
errors = sum(result != target)
print("Nb erreurs:", errors)
print( "Pourcentage de prédiction juste:", (150-errors)*100/150)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(result, target)
```

Partie 2

Au niveau de l'apprentissage non supervisé, il est très fréquent de disposer de très grandes quantités de paramètres. Vous allez tout livrer à la machine.

Cela pose 2 problèmes:

- La visualisation des données, au delà de 3 paramètres notre cerveau est bien mal outillé pour se représenter les données
- La complexité des calculs, plus le nombre de paramètres est grand, plus nous aurons des calculs complexes et longs

Pour contourner ces problèmes il est courant de **réduire la dimension** du vecteur de données à quelque chose de plus simple. La difficulté est alors de réduire le nombre de paramètres tout en conservant l'essentiel de l'information, notamment les variations succeptibles de permettre le regroupement des données.

Plusieurs techniques de réduction sont disponibles avec Scikit-Learn.

Dans ce TP Nous utiliserons pour l' analyse en composante principale, dite PCA. Le module manifold propose aussi d'autres types d'algorithmes.

PCA est une technique **linéaire** de réduction de dimension qui a l'avantage d'être très rapide.

- Vous définissez le nombre de paramètres
- Vous alimentez l'algorithme avec les données à réduire
- Vous lancez la prédiction ici appelée réduction/transformation

1. Expliquez ces lignes de code

```
from sklearn.decomposition import PCA

model = PCA(n_components=2)

model.fit(iris.data)

reduc = model.transform(iris.data )
```

……
……
……
……

2. Création du dataframe après la réduction et visualization des données (dans cette partie du TP vous allez travailler avec seaborn).

```
import seaborn as sns
import pandas as pd
sns.set()
df = pd.DataFrame(data, columns=iris['feature_names'] )
df['target'] = target
df['label'] = df.apply(lambda x: iris['target_names'][int(x.target)], axis=1)
df.head()
```

```
df['PCA1'] = reduc[:, 0]
df['PCA2'] = reduc[:, 1]
df.head()
```

```
sns.lmplot("PCA1", "PCA2", hue='label', data=df, fit_reg=False);
```

3. Lancement de l'apprentissage et la prédiction. Comparez les résultats de prédiction trouvés par l'apprentissage supervisé et non supervisé. Interprétez.

……
……
……
……
……..

```
from sklearn.mixture import GaussianMixture
# Création du modèle avec 3 groupes de données
model = GaussianMixture (n_components=3, covariance_type='full')

model.fit(df[['PCA1', 'PCA2']])
# Prédiction
groups = model.predict(df[['PCA1', 'PCA2']])

df['group'] = groups
sns.lmplot("PCA1", "PCA2", data=df, hue='label',
           col='group', fit_reg=False);

result = model.predict(df[['PCA1', 'PCA2']])
result
```

Compte Rendu :

Ecrire le code permettant de calculer le score de prédiction de ce training.

……
……