Année universitaire	**2021-2022**		
Département	**Informatique**	**Année**	**L3**
Matière	**LIFO65**		
Enseignant	**Khalid Benabdeslem & Haytham Elghazel & Mehdi Hennequin**		
Intitulé TD/TP :	**TP2 Clustering avec Python**		
Contenu	• **Clustering** • **Evaluation de la qualité d'un clustering**		

Pour ce TP, Il faut rendre votre notebook python sur claroline. Sous la forme TP2_LIFO65_Nom_Prenom.ipynb.

Dans ce TP, vous allez expérimenter des algorithmes de traitement de données pour répondre à différents problèmes liés au clustering avec le langage Python.

Pour plus de détails concernant :

- le langage Python vous pouvez aller sur le site suivant : http://www.python-course.eu/index.php
- la librairie Scikit-learn vous pouvez aller sur le site suivant : http://scikit-learn.org

Pour lancer le notebook Python, il faut taper la commande **jupyter notebook** dans votre dossier de travail. Une fenêtre va se lancer dans votre navigateur pour ouvrir l'application Jupyter. Créer un nouveau notebook Python et taper le code suivant dans une nouvelle cellule :

```
import numpy as np
np.set_printoptions(threshold=10000,suppress=True)
import pandas as pd
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
```

La classification (les Anglos-saxons parlent de clustering) est l'opération statistique qui consiste à regrouper des objets (individus ou variables) en un nombre limité de groupes, les classes (ou segments, ou clusters) , qui ont deux propriétés. D'une part, ils ne sont pas prédéfinis par l'analyste mais découverte au cours de l'opération, contrairement aux classes du classement. D'autre part, les classes de la classification regroupent les objets ayant des caractéristiques similaires et séparent les objets ayant des caractéristiques différentes (homogénéité interne et hétérogénéité externe), ce qui peut être mesuré par des critères telle l'inertie interclasse et l'inertie intraclasse.

Nous allons étudier dans la suite deux approches de clustering (*k-moyennes* **"KMeans"** et la *classification Ascendante Hiérarchique* **"AgglomerativeClustering"** du package sklearn.cluster) que nous allons appliquer sur le jeu de données des **villes**.

1. Appliquez la procédure **KMeans** sur ce jeu de données pour obtenir **3 clusters**
   - Donner une visualisation graphique des villes projetées dans le plan principal. Les villes de chaque cluster devraient avoir une couleur différente des villes des autres clusters (voir code ci-dessous).

   X_pca étant la matrice des données transformées par l'ACP, **labels** étant le vecteur contenant le nom des instances (ici les villes), **clustering** étant le clustering obtenu.

   ```
   colors = ['red','yellow','blue','pink']
   plt.scatter(X_pca[:, 0], X_pca[:, 1], c= clustering, cmap=matplotlib.colors.ListedColormap(colors))
   for label, x, y in zip(labels, X_pca[:, 0], X_pca[:, 1]):
       plt.annotate(label, xy=(x, y), xytext=(-0.2, 0.2), textcoords='offset points')
   plt.show()
   ```

2. Appliquez la procédure **AgglomerativeClustering** sur ce jeu de données pour obtenir **3 clusters** avec différentes méthodes d'agrégation (il faut essayer **ward**, **average** et **single**).
   - Donner à chaque fois une visualisation graphique des villes projetées dans le plan principal. Les

villes de chaque cluster devraient avoir une couleur différente des villes des autres clusters. Comparer en analysant les résultats visuels obtenus.

Nous allons maintenant déterminer la meilleure partition (nombre de clusters) pour les méthodes **KMeans** et **AgglomerativeClustering**. Pour cela, nous allons utiliser le critère **"Silhouette index"** (**metrics.silhouette_score** de scikit-learn).

3. Utiliser cet indice dans une boucle de 5 itérations au maximum. Les 5 itérations correspondent aux 5 partitions possibles *i.e.* en 2, 3, 4, 5 et 6 classes issues de **KMeans**. Déduire la meilleure partition qui correspond à un indice maximal pour l'indice **Silhouette**.
4. Utiliser cet indice dans une boucle de 5 itérations au maximum. Les 5 itérations correspondent aux 5 partitions possibles *i.e.* en 2, 3, 4, 5 et 6 classes issues de **AgglomerativeClustering** (avec à chaque le critère d'agrégation **ward**, **average** et **single**. Déduire la meilleure partition qui correspond à un indice maximal pour l'indice **Silhouette**.
5. Si nous souhaitons avoir 3 clusters, quelle méthode (entre les 4 testées précédemment) donnera la meilleure partition en 3 clusters ?
6. Citer les avantages et les inconvénients des méthodes de classification hiérarchiques (**AgglomerativeClustering**) et celles de partitionnement (**KMeans**).
7. Proposer dans une fonction une éventuelle approche permettant de combiner les points forts des méthodes hiérarchiques et des méthodes de partitionnement : on appelle une telle approche **mixte** ou **hybride**. Proposer une visualisation du nuage des villes et comparer les résultats obtenus avec ceux des questions précédentes.
8. Appliquer la fonction précédente sur les jeux de données **"wdbc.csv"** et **"spamb.csv"**.