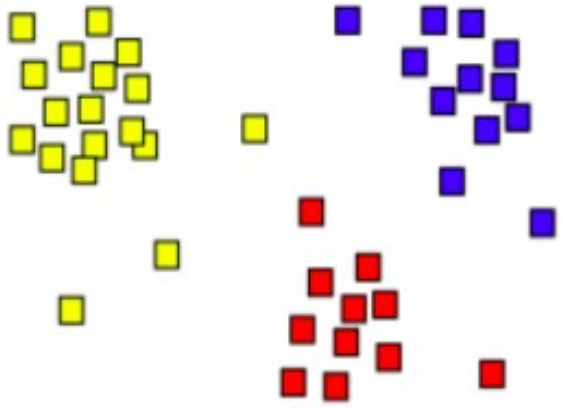
Université Abdelmalek Essaad Faculté ses Sciences et techniques de Tanger
Département Génie Informatique

Atelier 3
«Clustering»



Réalisée par :
KAISSI Houda

Objective :
l'objective principal de cet atelier est de pratiquer les concepts du clustering, en
traitant les données d'une Data Sets.

Outils :
Python , Pandas, Sklearn, matplotlib.

Partie 1 (Data Visualisation):

1. En utilisant pandas essayer d'explorer les données du Data set.

```
import pandas as pd

# Chemin d'accès au Dataset
url = "CC GENERAL.csv"

# Lire le fichier
dataset = pd.read_csv(url)

# Affichage de la première partie du dataset
print("----- Head -----")
print(dataset.head())

# Affichage du nombre de lignes et de colonnes du dataset
print("----- Shape -----")
print(dataset.shape)
```

Resultat :

```
----- Head -----
  CUST_ID      BALANCE  BALANCE_FREQUENCY  PURCHASES  ONEOFF_PURCHASES  \
0  C10001    40.900749           0.818182      95.40              0.00
1  C10002  3202.467416           0.909091       0.00              0.00
2  C10003  2495.148862           1.000000     773.17            773.17
3  C10004  1666.670542           0.636364    1499.00           1499.00
4  C10005   817.714335           1.000000      16.00             16.00

   INSTALLMENTS_PURCHASES  CASH_ADVANCE  PURCHASES_FREQUENCY  \
0                    95.4      0.000000             0.166667
1                     0.0   6442.945483             0.000000
2                     0.0      0.000000             1.000000
3                     0.0    205.788017             0.083333
4                     0.0      0.000000             0.083333

   ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
0                    0.000000                          0.083333
1                    0.000000                          0.000000
2                    1.000000                          0.000000
3                    0.083333                          0.000000
4                    0.083333                          0.000000

   CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
0                0.000000                 0              2        1000.0
1                0.250000                 4              0        7000.0
2                0.000000                 0             12        7500.0
3                0.083333                 1              1        7500.0
4                0.000000                 0              1        1200.0

      PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT  TENURE
0   201.802084        139.509787          0.000000      12
1  4103.032597       1072.340217          0.222222      12
```

2. Afficher le résumer statistique du Data Sets avec une interprétation des résultats obtenues.

```
:
```

```python
import pandas as pd

# Chemin d'accès au Dataset
url = "CC GENERAL.csv"

# Lire le fichier
dataset = pd.read_csv(url)

# Résumé statistique
print("----- Résumé statistique -----")
print(dataset.describe())
```

Resultat :

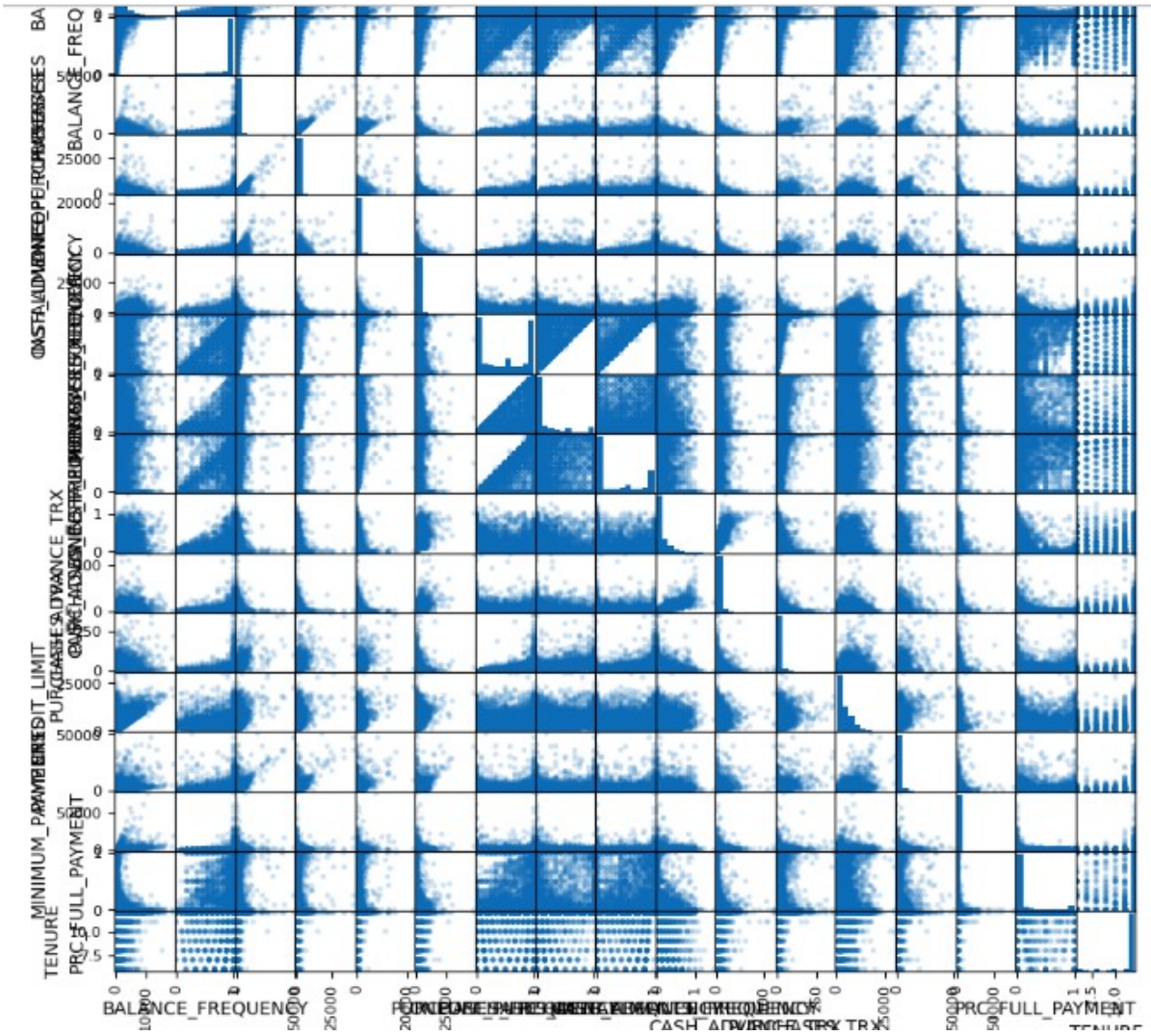
```
----- Résumé statistique -----
            BALANCE  BALANCE_FREQUENCY     PURCHASES  ONEOFF_PURCHASES  \
count   8950.000000        8950.000000   8950.000000       8950.000000
mean    1564.474828           0.877271   1003.204834        592.437371
std     2081.531879           0.236904   2136.634782       1659.887917
min        0.000000           0.000000      0.000000          0.000000
25%      128.281915           0.888889     39.635000          0.000000
50%      873.385231           1.000000    361.280000         38.000000
75%     2054.140036           1.000000   1110.130000        577.405000
max    19043.138560           1.000000  49039.570000      40761.250000

       INSTALLMENTS_PURCHASES  CASH_ADVANCE  PURCHASES_FREQUENCY  \
count             8950.000000   8950.000000          8950.000000
mean               411.067645    978.871112             0.490351
std                904.338115   2097.163877             0.401371
min                  0.000000      0.000000             0.000000
25%                  0.000000      0.000000             0.083333
50%                 89.000000      0.000000             0.500000
75%                468.637500   1113.821139             0.916667
max              22500.000000  47137.211760             1.000000

       ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
count                 8950.000000                       8950.000000
mean                     0.202458                          0.364437
std                      0.298336                          0.397448
min                      0.000000                          0.000000
25%                      0.000000                          0.000000
50%                      0.083333                          0.166667
75%                      0.300000                          0.750000
max                      1.000000                          1.000000


        CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
count             8950.000000       8950.000000    8950.000000   8949.000000
mean                 0.135144          3.248827      14.709832   4494.449450
std                  0.200121          6.824647      24.857649   3638.815725
min                  0.000000          0.000000       0.000000     50.000000
25%                  0.000000          0.000000       1.000000   1600.000000
50%                  0.000000          0.000000       7.000000   3000.000000
75%                  0.222222          4.000000      17.000000   6500.000000
max                  1.500000        123.000000     358.000000  30000.000000

            PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT       TENURE
count   8950.000000       8637.000000       8950.000000  8950.000000
mean    1733.143852        864.206542          0.153715    11.517318
std     2895.063757       2372.446607          0.292499     1.338331
min        0.000000          0.019163          0.000000     6.000000
25%      383.276166        169.123707          0.000000    12.000000
50%      856.901546        312.343947          0.000000    12.000000
75%     1901.134317        825.485459          0.142857    12.000000
max    50721.483360      76406.207520          1.000000    12.000000
```

3. Afficher les nuages des points du data set selon les propriétés « Features » en utilisant matplotlib et
pandas « scatter_matrix ».

```python
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

# Chemin d'accès au Dataset
url = "CC GENERAL.csv"

# Lire le fichier
dataset = pd.read_csv(url)

# Résumé statistique
print("----- Résumé statistique -----")
print(dataset.describe())

# Afficher les nuages de points (scatter_matrix)
scatter_matrix(dataset, alpha=0.2, figsize=(10, 10), diagonal='hist')
plt.show()
```

Resultat :

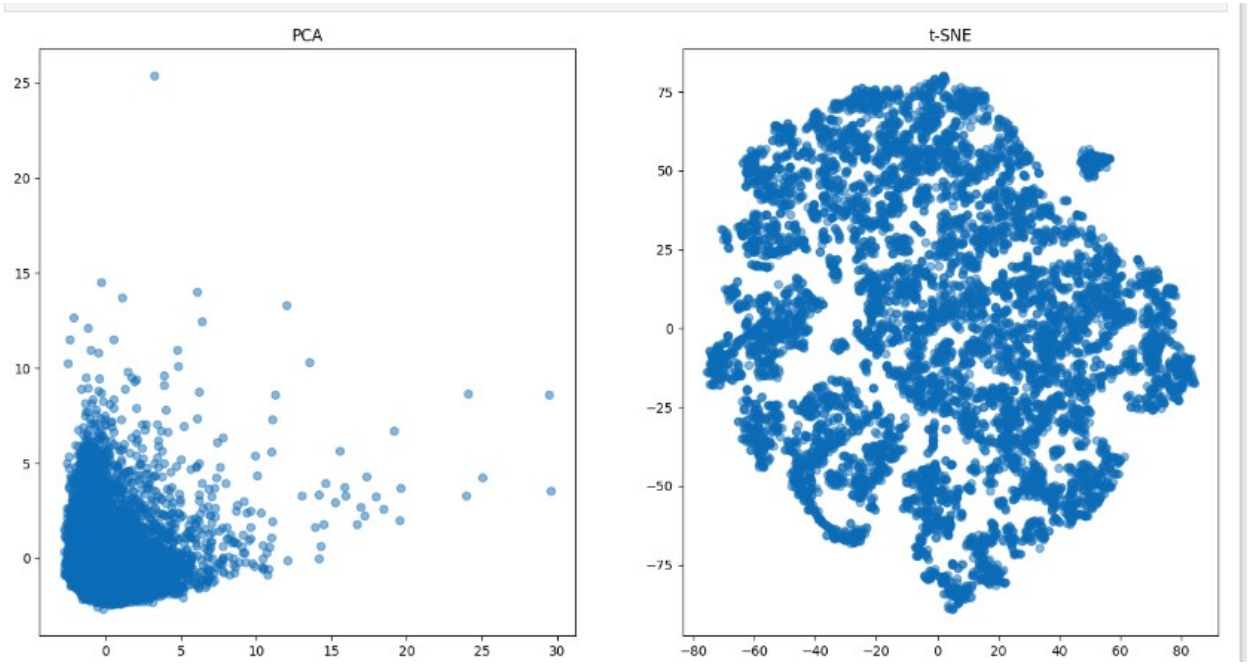
```
----- Résumé statistique -----
            BALANCE  BALANCE_FREQUENCY     PURCHASES  ONEOFF_PURCHASES  \
count   8950.000000        8950.000000   8950.000000       8950.000000
mean    1564.474828           0.877271   1003.204834        592.437371
std     2081.531879           0.236904   2136.634782       1659.887917
min        0.000000           0.000000      0.000000          0.000000
25%      128.281915           0.888889     39.635000          0.000000
50%      873.385231           1.000000    361.280000         38.000000
75%     2054.140036           1.000000   1110.130000        577.405000
max    19043.138560           1.000000  49039.570000      40761.250000

       INSTALLMENTS_PURCHASES  CASH_ADVANCE  PURCHASES_FREQUENCY  \
count             8950.000000   8950.000000          8950.000000
mean               411.067645    978.871112             0.490351
std                904.338115   2097.163877             0.401371
min                  0.000000      0.000000             0.000000
25%                  0.000000      0.000000             0.083333
50%                 89.000000      0.000000             0.500000
75%                468.637500   1113.821139             0.916667
max              22500.000000  47137.211760             1.000000

       ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
count                 8950.000000                       8950.000000
mean                     0.202458                          0.364437
std                      0.298336                          0.397448
min                      0.000000                          0.000000
25%                      0.000000                          0.000000
50%                      0.083333                          0.166667
75%                      0.300000                          0.750000
max                      1.000000                          1.000000

       CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
count             8950.000000       8950.000000    8950.000000   8949.000000
mean                 0.135144          3.248827      14.709832   4494.449450
std                  0.200121          6.824647      24.857649   3638.815725
min                  0.000000          0.000000       0.000000     50.000000
25%                  0.000000          0.000000       1.000000   1600.000000
50%                  0.000000          0.000000       7.000000   3000.000000
75%                  0.222222          4.000000      17.000000   6500.000000
max                  1.500000        123.000000     358.000000  30000.000000
```



4. Appliquer les deux techniques PCA et Tsne sur les features du Dataset, que ce que vous constatez.

```python
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Chemin d'accès au Dataset
url = "CC GENERAL.csv"

# Lire le fichier
dataset = pd.read_csv(url)

# Supprimer les colonnes non numériques (si nécessaire)
dataset_numeric = dataset.select_dtypes(include=['number'])

# Remplir les valeurs manquantes (si nécessaire)
dataset_numeric = dataset_numeric.fillna(dataset_numeric.mean())

# Standardiser les données (si nécessaire)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
dataset_scaled = scaler.fit_transform(dataset_numeric)

# Appliquer PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(dataset_scaled)

# Appliquer t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_result = tsne.fit_transform(dataset_scaled)

# Visualiser les résultats
plt.figure(figsize=(16, 8))

plt.subplot(1, 2, 1)
plt.scatter(pca_result[:, 0], pca_result[:, 1], alpha=0.5)
plt.title('PCA')

plt.subplot(1, 2, 2)
plt.scatter(tsne_result[:, 0], tsne_result[:, 1], alpha=0.5)
plt.title('t-SNE')

plt.show()
```

Resultat :



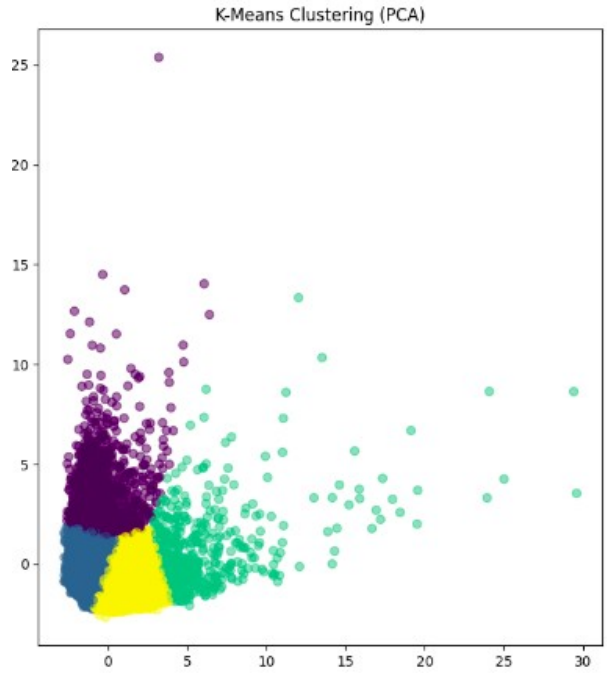
Partie 2 (Clustering):

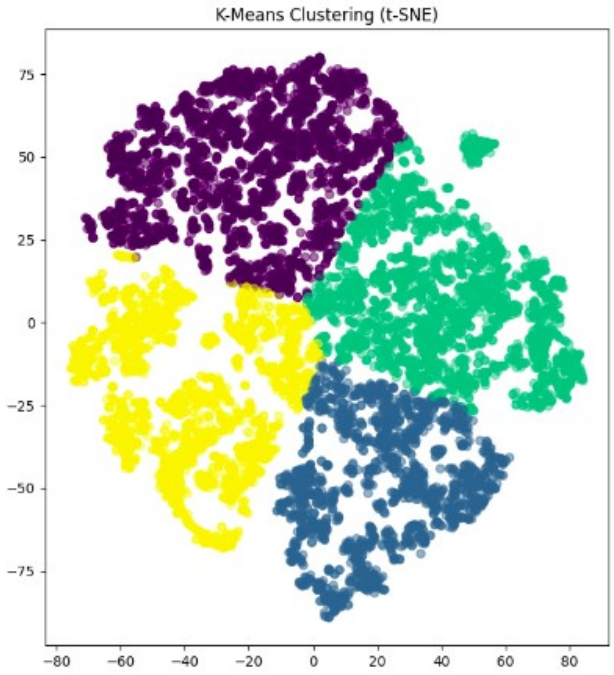
1. Essayer de construire les modèles de clustering en utilisant Kmeans (avec les nouvelles
features (Un modèle basè sur PCA et l'autre sur Tsne) Question 4 de la partie 1.

```
]: import pandas as pd
   from sklearn.cluster import KMeans
   import matplotlib.pyplot as plt
   from sklearn.decomposition import PCA
   from sklearn.manifold import TSNE
   from sklearn.preprocessing import StandardScaler

   # Charger le dataset
   url = "CC GENERAL.csv"
   dataset = pd.read_csv(url)

   # Prétraitement des données
   # Supprimer les colonnes non numériques (si nécessaire)
   dataset_numeric = dataset.select_dtypes(include=['number'])
   # Remplir les valeurs manquantes (si nécessaire)
   dataset_numeric = dataset_numeric.fillna(dataset_numeric.mean())

   # Standardiser les données
   scaler = StandardScaler()
   dataset_scaled = scaler.fit_transform(dataset_numeric)

   # Appliquer PCA
   pca = PCA(n_components=2)
   pca_result = pca.fit_transform(dataset_scaled)

   # Appliquer t-SNE
   tsne = TSNE(n_components=2, random_state=42)
   tsne_result = tsne.fit_transform(dataset_scaled)

   # Construire le modèle de clustering avec K-Means pour PCA
   kmeans_pca = KMeans(n_clusters=4, random_state=42)
   dataset['Cluster_PCA'] = kmeans_pca.fit_predict(pca_result)
```

```
# Construire le modèle de clustering avec K-Means pour t-SNE
kmeans_tsne = KMeans(n_clusters=4, random_state=42)
dataset['Cluster_TSNE'] = kmeans_tsne.fit_predict(tsne_result)

# Visualiser les résultats pour PCA
plt.figure(figsize=(16, 8))

plt.subplot(1, 2, 1)
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=dataset['Cluster_PCA'], cmap='viridis', alpha=0.5)
plt.title('K-Means Clustering (PCA)')

# Visualiser les résultats pour t-SNE
plt.subplot(1, 2, 2)
plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=dataset['Cluster_TSNE'], cmap='viridis', alpha=0.5)
plt.title('K-Means Clustering (t-SNE)')

plt.show()
```

Resultat :

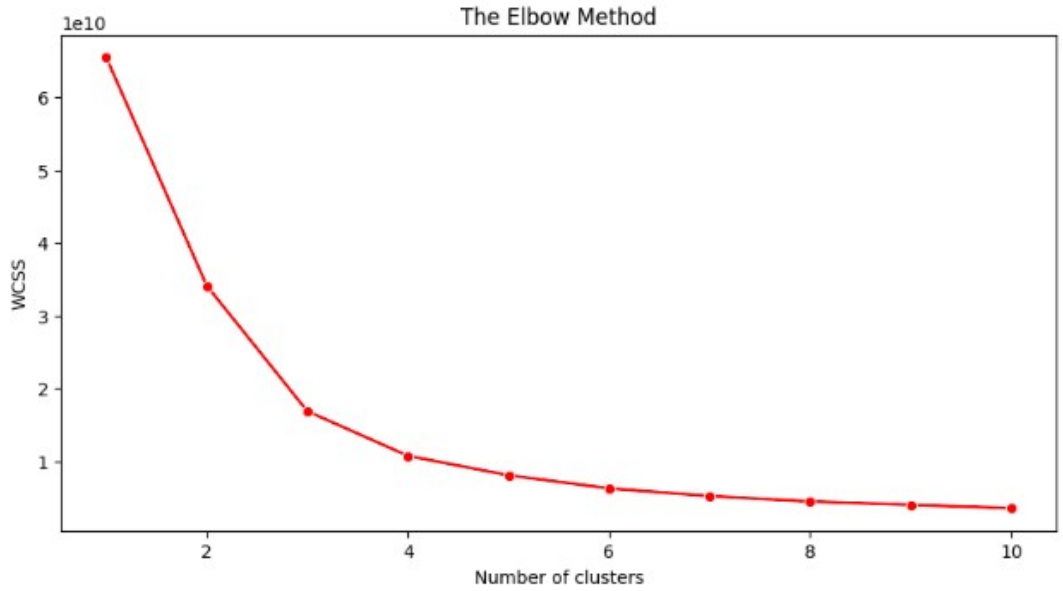




2. Définir le K nécessaire pour les deux modèles en utilisant la méthode d’Elbow.

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# Importation du dataset
dataset = pd.read_csv("CC GENERAL.csv", index_col='CUST_ID')
dataset.drop_duplicates(inplace=True)

# Pour la visualisation, on utilise juste les variables BALANCE_FREQUENCY et PURCHASES
X = dataset.iloc[:, [2, 3]].values

# Application de la méthode de Elbow pour trouver le nombre optimal de clusters
wcss = []

for i in range(1, 11):  # de 1 à 10 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    # inertia method returns wcss for that model
    wcss.append(kmeans.inertia_)  # On ajoute à chaque fois au tableau le cluster qu'on a

plt.figure(figsize=(10, 5))
sns.lineplot(x=range(1, 11), y=wcss, marker='o', color='red')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Resultat :

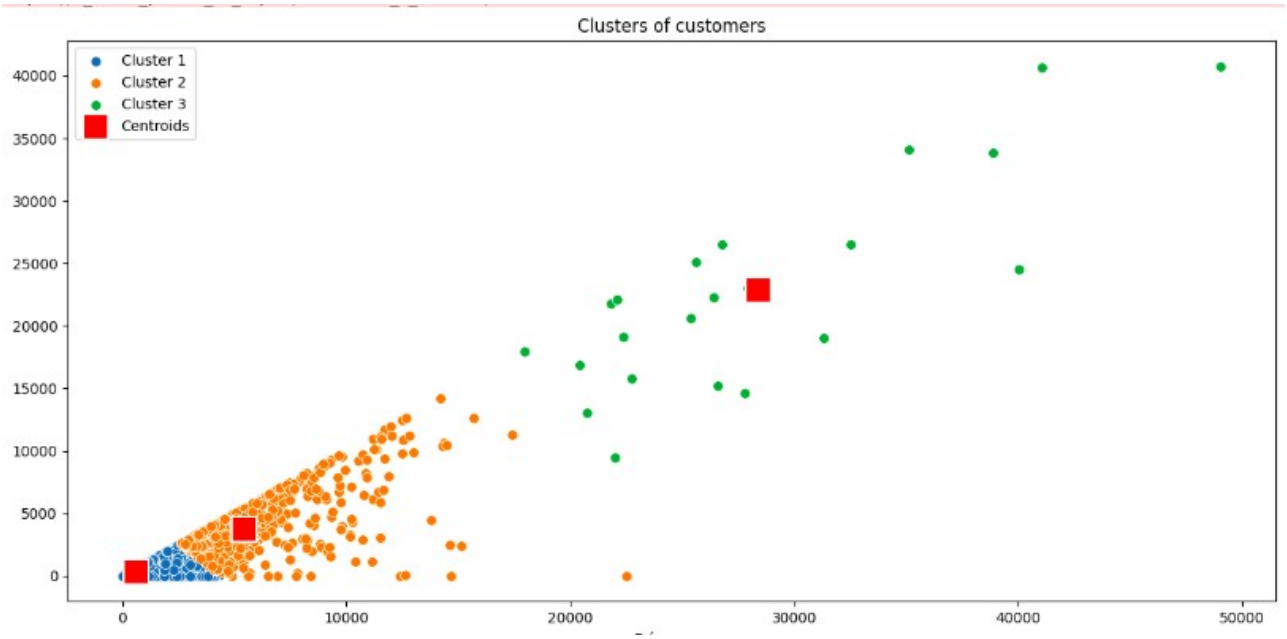


3. Présenter les clusters obtenues dans un graphe en utilisant matplotlib.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

# Importation du dataset
dataset = pd.read_csv("CC GENERAL.csv", index_col='CUST_ID')
dataset.drop_duplicates(inplace=True)

# Pour la visualisation, on utilise juste les variables BALANCE_FREQUENCY et PURCHASES
X = dataset.iloc[:, [2, 3]].values

# Application de la méthode de Elbow pour trouver le nombre optimal de clusters
wcss = []

for i in range(1, 11):  # de 1 à 10 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    # inertia method returns wcss for that model
    wcss.append(kmeans.inertia_)  # On ajoute à chaque fois au tableau le cluster qu'on a

# Affichage des valeurs wcss pour chaque cluster
# Utilisation de la méthode de coude pour trouver le nombre optimal de clusters
plt.figure(figsize=(10, 5))
```

```
# Visualisation des clusters
plt.figure(figsize=(15, 7))
for cluster_num in range(optimal_clusters):
    sns.scatterplot(x=X[y_kmeans == cluster_num, 0], y=X[y_kmeans == cluster_num, 1], label=f'Cluster {cluster_num + 1}', s=5

sns.scatterplot(x=kmeans.cluster_centers_[:, 0], y=kmeans.cluster_centers_[:, 1], color='red', label='Centroids', s=300, mark
plt.grid(False)
plt.title('Clusters of customers')
plt.xlabel('Fréquence')
plt.ylabel('Montant des achats')
plt.legend()
plt.show()
```

Resultat :



4.Interpréter les résultats obtenus des deux modèles .

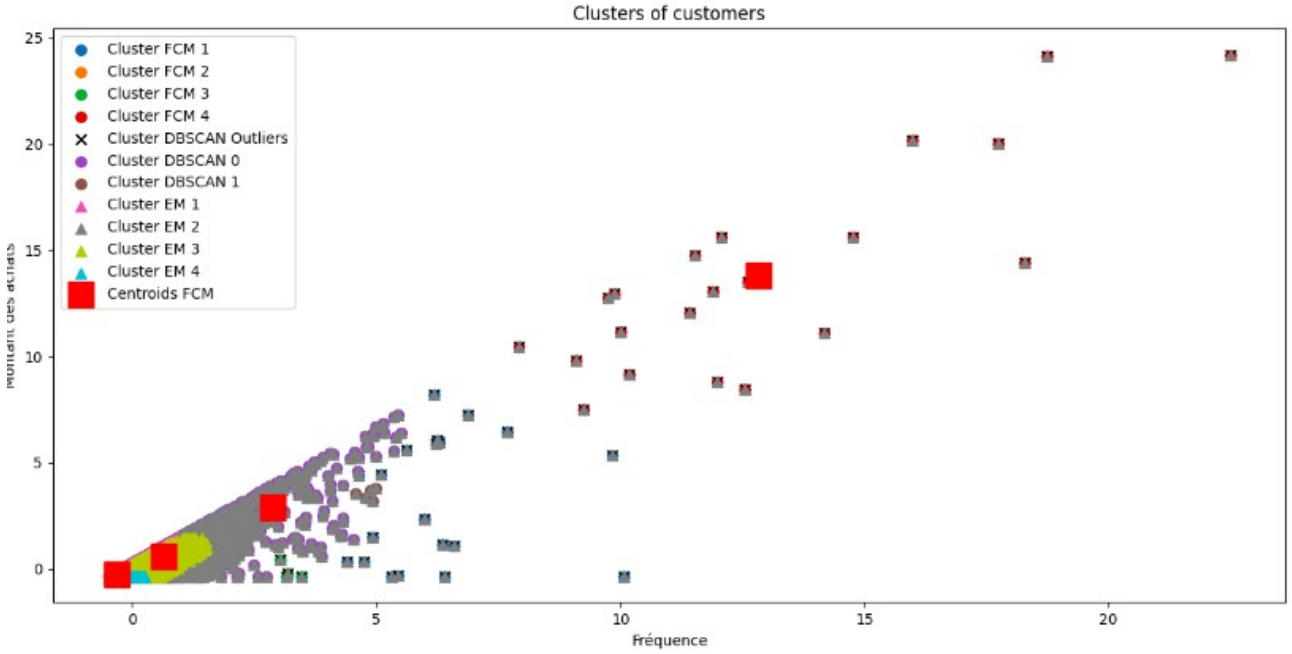
Pour PCA Les clusters formés semblent relativement bien séparés, mais il y a une certaine chevauchement entre eux. Les données ont été projetées sur les deux principales composantes principales.
Contrairement à PCA, t-SNE capture des relations non linéaires entre les points. Les clusters semblent mieux séparés, avec moins de chevauchement entre eux.

5. Refaire la même chose en utilisant l’algorithme fuzzy cmeans « il faut utiliser la bibliothèque skfuzzy
» DBSCAN , EM , SOM et Hierarchical clustering .

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import skfuzzy as fuzz
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
from sklearn.mixture import GaussianMixture

# Importation du dataset
dataset = pd.read_csv("CC GENERAL.csv", index_col='CUST_ID')
dataset.drop_duplicates(inplace=True)

# Sélection des variables pour clustering
X = dataset.iloc[:, [2, 3]].values

# Normalisation des données
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Application de l'algorithme Fuzzy C-means
n_clusters = 4  # Remplacez ceci par le nombre de clusters souhaité
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(X_scaled.T, n_clusters, 2, error=0.005, maxiter=1000)

# Attribution des clusters FCM
cluster_membership_fcm = np.argmax(u, axis=0)

# Application de l'algorithme DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
cluster_membership_dbscan = dbscan.fit_predict(X_scaled)

# Application de l'algorithme EM (Gaussian Mixture Model)
gmm = GaussianMixture(n_components=n_clusters)
cluster_membership_em = gmm.fit_predict(X_scaled)

# Visualisation des clusters obtenus
plt.figure(figsize=(15, 7))

# FCM
for i in range(n_clusters):
    plt.scatter(X_scaled[cluster_membership_fcm == i, 0], X_scaled[cluster_membership_fcm == i, 1],
                label=f'Cluster FCM {i + 1}', s=50)
```

```python
# DBSCAN
plt.scatter(X_scaled[cluster_membership_dbscan == -1, 0], X_scaled[cluster_membership_dbscan == -1, 1],
            label='Cluster DBSCAN Outliers', s=50, color='black', marker='x')
for i in np.unique(cluster_membership_dbscan[cluster_membership_dbscan != -1]):
    plt.scatter(X_scaled[cluster_membership_dbscan == i, 0], X_scaled[cluster_membership_dbscan == i, 1],
                label=f'Cluster DBSCAN {i}', s=50)

# EM
for i in range(n_clusters):
    plt.scatter(X_scaled[cluster_membership_em == i, 0], X_scaled[cluster_membership_em == i, 1],
                label=f'Cluster EM {i + 1}', s=50, marker='^')

plt.scatter(cntr[:, 0], cntr[:, 1], color='red', label='Centroids FCM', s=300, marker=',')
plt.grid(False)
plt.title('Clusters of customers')
plt.xlabel('Fréquence')
plt.ylabel('Montant des achats')
plt.legend()
plt.show()
```

Resultat :



6. Comparer les cinq algorithmes et faire une conclusion globale sur ces méthodes de clustering.

K-Means est simple mais peut être limité par la forme des clusters

Fuzzy C-means est utile lorsque des points peuvent appartenir à plusieurs clusters.

DBSCAN est robuste pour des formes de clusters variées et résistant aux outliers.

EM est adapté à des modèles probabilistes complexes.

SOM est puissant pour les structures non linéaires et la visualisation topologique