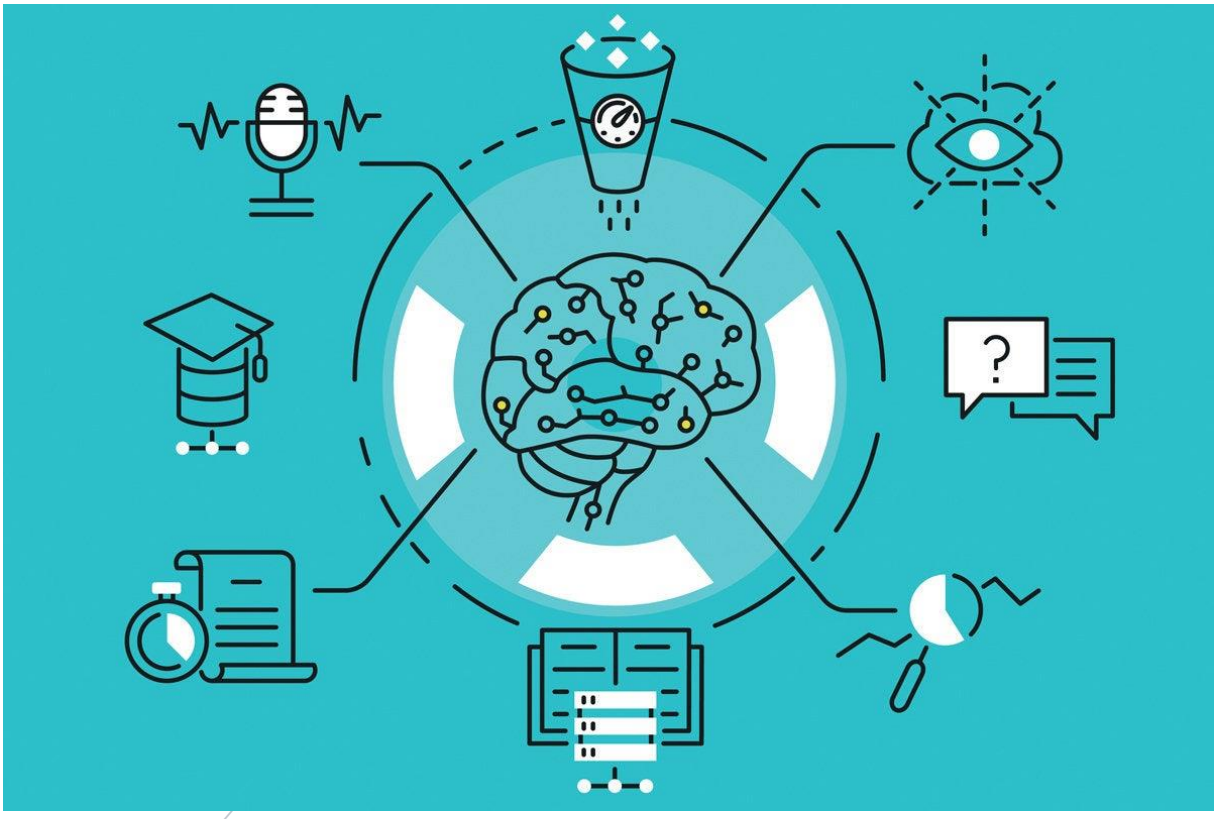
Université Abdelmalek Essaâdi
Faculté des Sciences et Techniques-Tanger
Département Génie Informatique

Machine Learning (Master SIBD)

Atelier 3



Réalisé Par :
Yossra safi chetouan

Encadre Par :
Pr . EL AACHAk LOTFI

Partie 1 (Data Visualisation):

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

[2]:
```
data=pd.read_csv("C:/dataset/CC GENERAL.csv")
```

[3]:
```
data
```

[3]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_F
0	C10001	40.900749	0.818182	95.40	0.00	95.40	0.000000	0.166667	
1	C10002	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	0.000000	
2	C10003	2495.148862	1.000000	773.17	773.17	0.00	0.000000	1.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.00	205.788017	0.083333	
4	C10005	817.714335	1.000000	16.00	16.00	0.00	0.000000	0.083333	
...	...	...	...	...	...	...	...	...	
8945	C19186	28.493517	1.000000	291.12	0.00	291.12	0.000000	1.000000	
8946	C19187	19.183215	1.000000	300.00	0.00	300.00	0.000000	1.000000	
8947	C19188	23.398673	0.833333	144.40	0.00	144.40	0.000000	0.833333	
8948	C19189	13.457564	0.833333	0.00	0.00	0.00	36.558778	0.000000	
8949	C19190	372.708075	0.666667	1093.25	1093.25	0.00	127.040008	0.666667	

8950 rows × 18 columns

[59]:
```
# Afficher les premières lignes du dataset
data.head()
```

[59]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PU
0	0	40.900749	0.990775	95.40	0.000000	95.4	0.000000	0.166667	
1	1	817.068179	0.909091	0.00	0.000000	0.0	228.525872	0.000000	
2	2	2495.148862	1.000000	773.17	773.170000	0.0	0.000000	1.000000	
3	3	1666.670542	0.990775	1499.00	128.659124	0.0	205.788017	0.083333	
4	4	817.714335	1.000000	16.00	16.000000	0.0	0.000000	0.083333	

[5]:
```
print(data.duplicated().sum())
```

```
0
```

[6]:
```
print(data.dtypes)
```

```
CUST_ID                            object
BALANCE                           float64
BALANCE_FREQUENCY                 float64
PURCHASES                         float64
ONEOFF_PURCHASES                  float64
INSTALLMENTS_PURCHASES            float64
CASH_ADVANCE                      float64
PURCHASES_FREQUENCY               float64
ONEOFF_PURCHASES_FREQUENCY        float64
PURCHASES_INSTALLMENTS_FREQUENCY  float64
CASH_ADVANCE_FREQUENCY            float64
CASH_ADVANCE_TRX                    int64
PURCHASES_TRX                       int64
CREDIT_LIMIT                      float64
PAYMENTS                          float64
MINIMUM_PAYMENTS                  float64
PRC_FULL_PAYMENT                  float64
TENURE                              int64
dtype: object
```

[7]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data['CUST_ID'] = label_encoder.fit_transform(data['CUST_ID'])
```

[8]:
```
null_counts = data.isnull().sum()
print(null_counts)
```

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

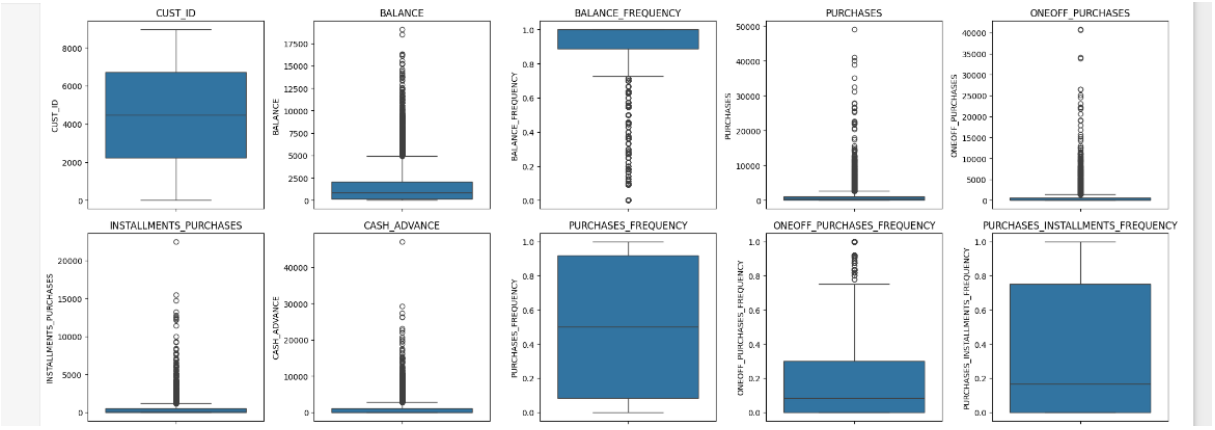
[9]:
```
mean_purchases_trx = data['PURCHASES_TRX'].mean()
data['PURCHASES_TRX'].fillna(mean_purchases_trx, inplace=True)
mean_credit_limit = data['CREDIT_LIMIT'].mean()
data['CREDIT_LIMIT'].fillna(mean_credit_limit, inplace=True)
mean_minimum_payments = data['MINIMUM_PAYMENTS'].mean()
data['MINIMUM_PAYMENTS'].fillna(mean_minimum_payments, inplace=True)
```

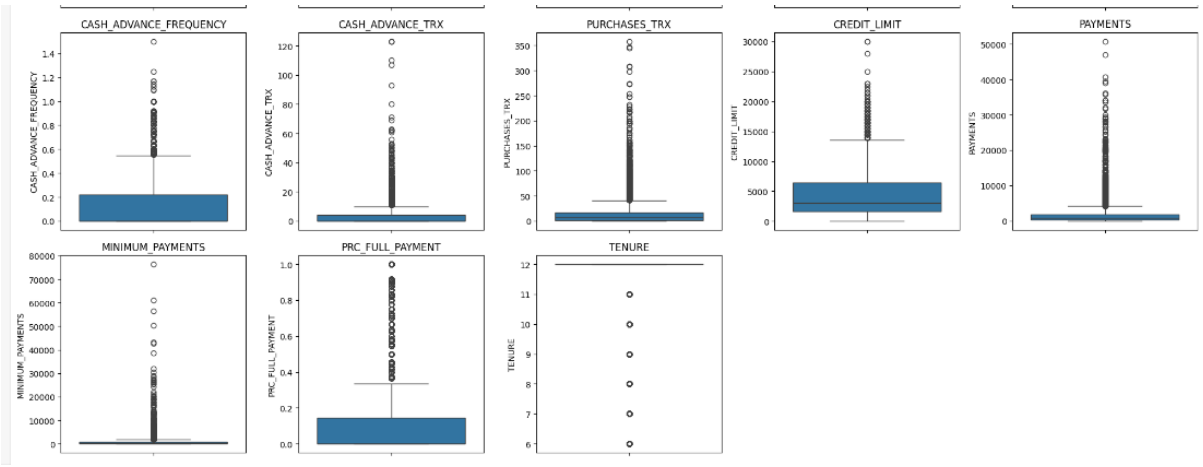
[10]:
```
data
```

[10]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF
0	0	40.900749	0.818182	95.40	0.00	95.40	0.000000	0.166667	
1	1	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	0.000000	
2	2	2495.148862	1.000000	773.17	773.17	0.00	0.000000	1.000000	
3	3	1666.670542	0.636364	1499.00	1499.00	0.00	205.788017	0.083333	
4	4	817.714335	1.000000	16.00	16.00	0.00	0.000000	0.083333	
...	...	...	...	...	...	...	...	...	
8945	8945	28.493517	1.000000	291.12	0.00	291.12	0.000000	1.000000	
8946	8946	19.183215	1.000000	300.00	0.00	300.00	0.000000	1.000000	
8947	8947	23.398673	0.833333	144.40	0.00	144.40	0.000000	0.833333	
8948	8948	13.457564	0.833333	0.00	0.00	0.00	36.558778	0.000000	

Afficher les valeurs aberrantes





Traitement les valeurs aberrantes

```
# Définition de la fonction pour remplacer les valeurs aberrantes par la moyenne
def replace_outliers_with_mean(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1

    # Définition des bornes pour les outliers
    lower_bound = Q1 - 0.5 * IQR
    upper_bound = Q3 + 0.5 * IQR

    # Filtrage des outliers
    column_filtered = column[(column >= lower_bound) & (column <= upper_bound)]

    # Remplacement des outliers par la moyenne de la colonne
    column_mean = column_filtered.mean()

    # Création d'une copie explicite pour éviter l'avertissement SettingWithCopyWarning
    column_copy = column.copy()

    # Remplacement des valeurs aberrantes
    column_copy.loc[column_copy < lower_bound] = column_mean
    column_copy.loc[column_copy > upper_bound] = column_mean

    return column_copy

# Appliquer la fonction à chaque colonne contenant des valeurs aberrantes
columns_with_outliers = [
'BALANCE',
'BALANCE_FREQUENCY',
'PURCHASES',
'ONEOFF_PURCHASES',
'INSTALLMENTS_PURCHASES',
'CASH_ADVANCE',
'PURCHASES_FREQUENCY',
```

```
'INSTALLMENTS_PURCHASES',
'CASH_ADVANCE',
'PURCHASES_FREQUENCY',
'ONEOFF_PURCHASES_FREQUENCY',
'CASH_ADVANCE_FREQUENCY',
'CASH_ADVANCE_TRX',
'CREDIT_LIMIT',
'PAYMENTS',
'MINIMUM_PAYMENTS',
'PRC_FULL_PAYMENT',
'TENURE',

    ]

for column_name in columns_with_outliers:
    data[column_name] = replace_outliers_with_mean(data[column_name])

print(data)
```

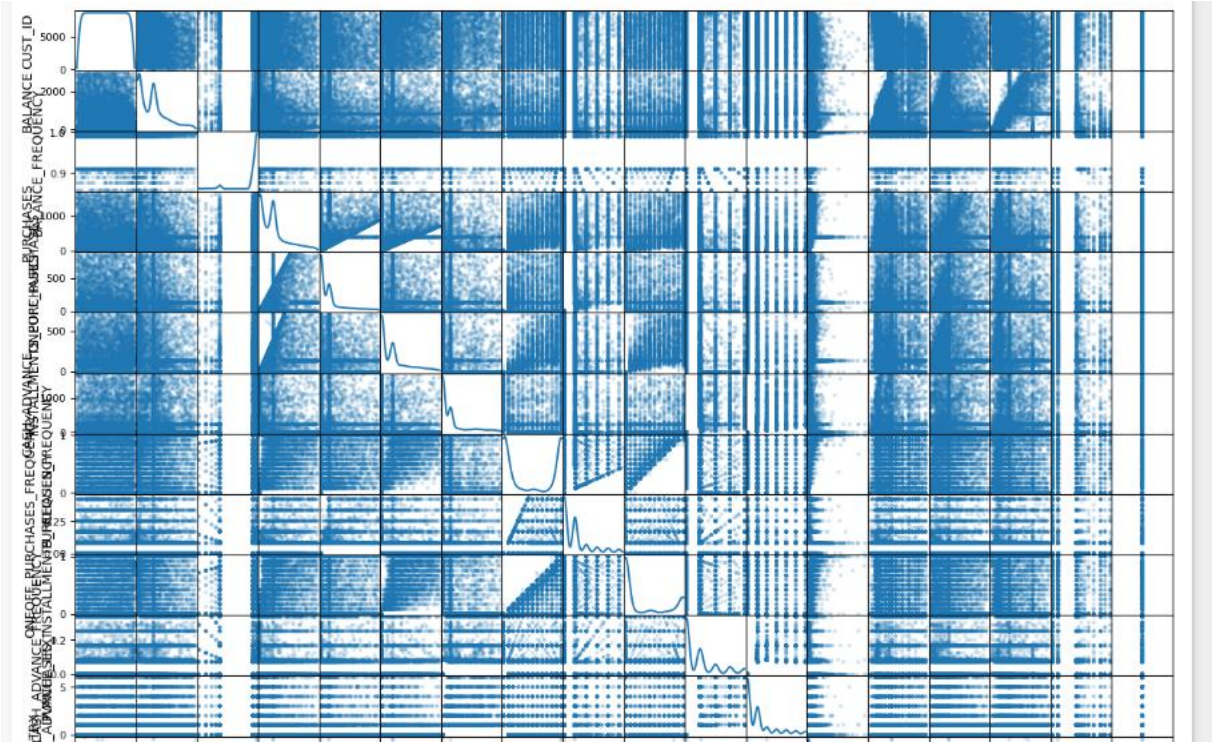
2. Afficher le résumer statistique du Data

```
[13]: # Afficher le résumé statistique du dataset
      data.describe()
```

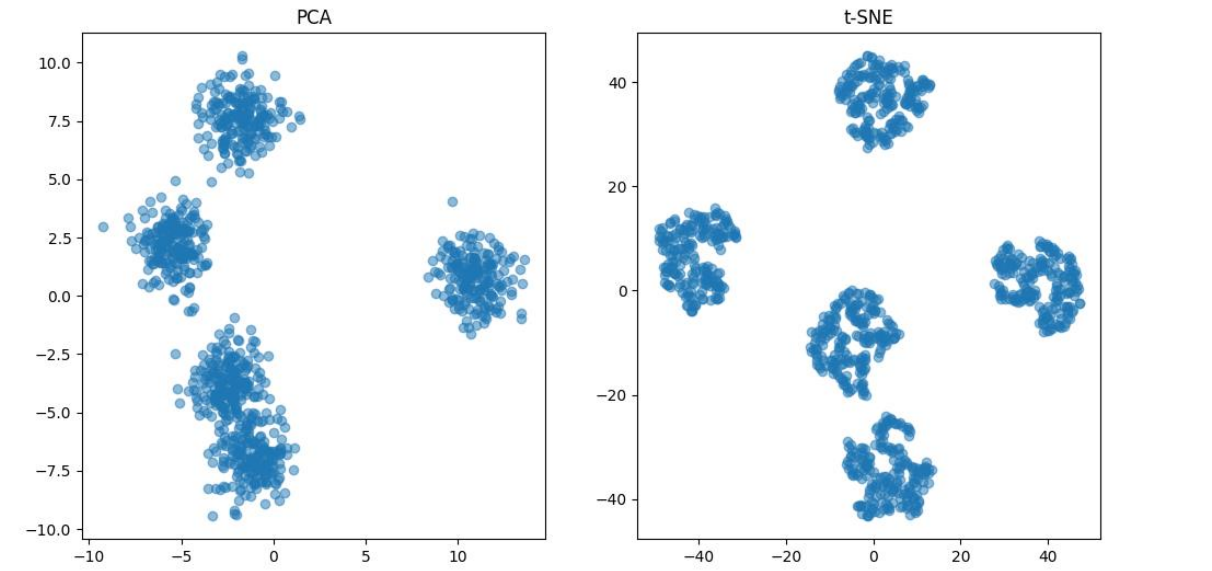
[13]:

ICY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	F
000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	
354	0.364437	0.072845	1.063657	14.709832	3425.900529	850.695140	378.636707	
279	0.397448	0.097485	1.549841	24.857649	2050.738370	602.732759	274.364762	
000	0.000000	0.000000	0.000000	0.000000	50.000000	0.000000	0.019163	
000	0.000000	0.000000	0.000000	1.000000	1600.000000	383.276166	170.857654	
354	0.166667	0.000000	0.000000	7.000000	3000.000000	850.695140	335.628312	
333	0.750000	0.083333	1.063657	17.000000	4500.000000	1110.862772	478.016728	
444	1.000000	0.333333	6.000000	358.000000	8800.000000	2658.629277	1210.778254	

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

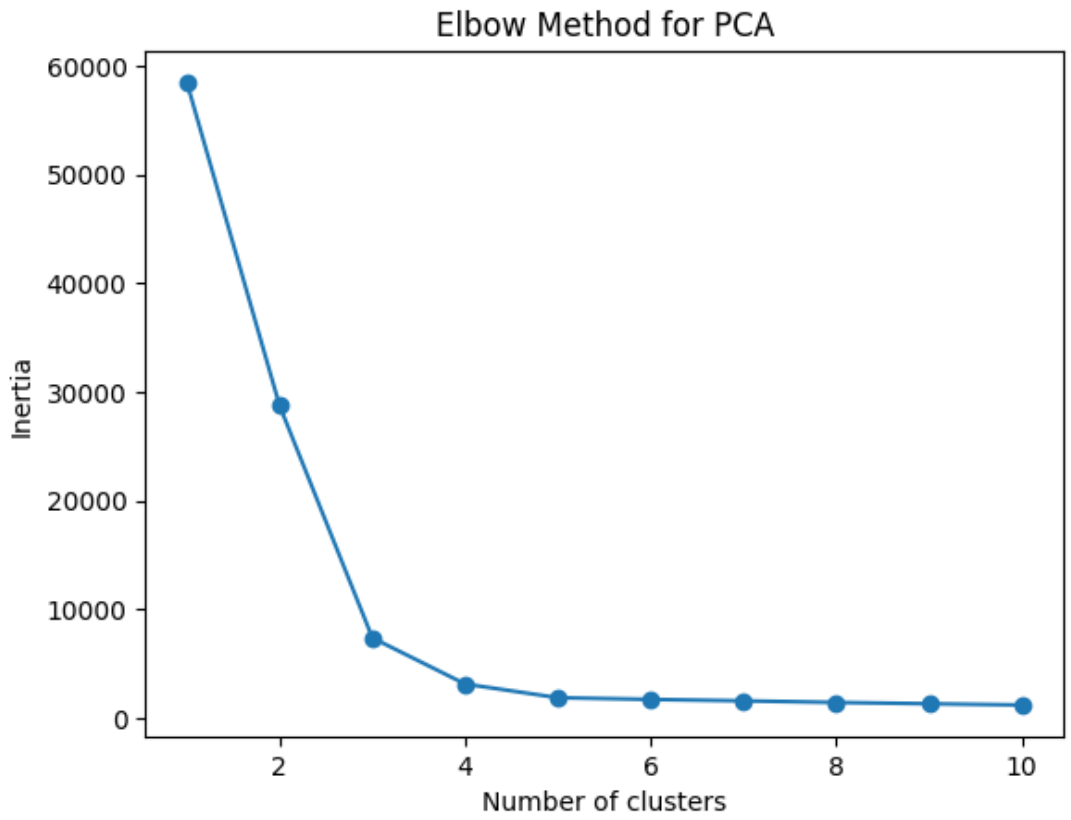


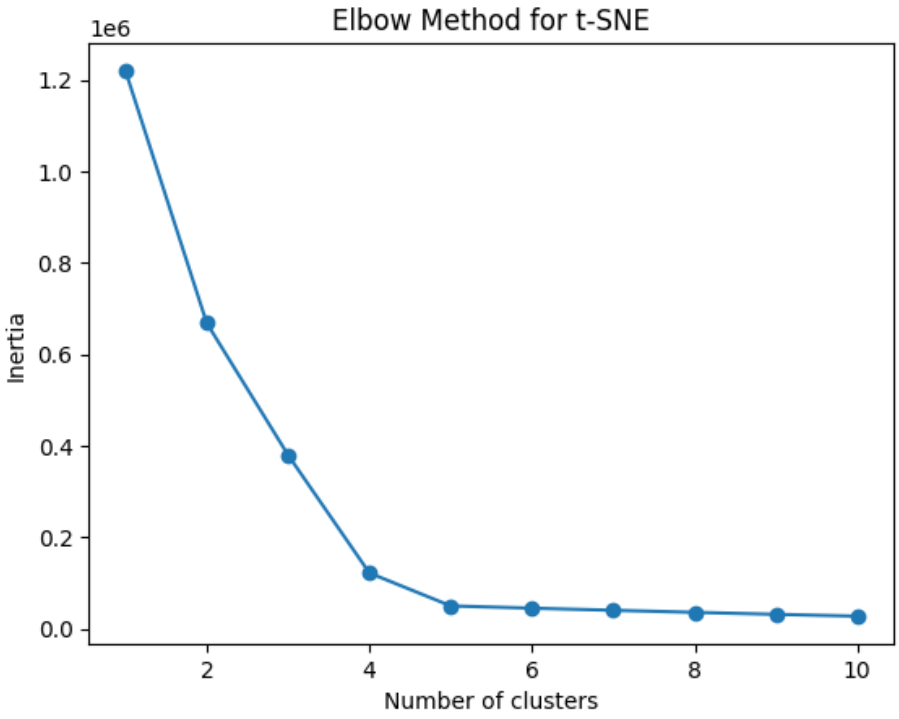
4. Appliquer les deux techniques PCA et Tsne sur les features du Dataset, que ce que vous constatez.



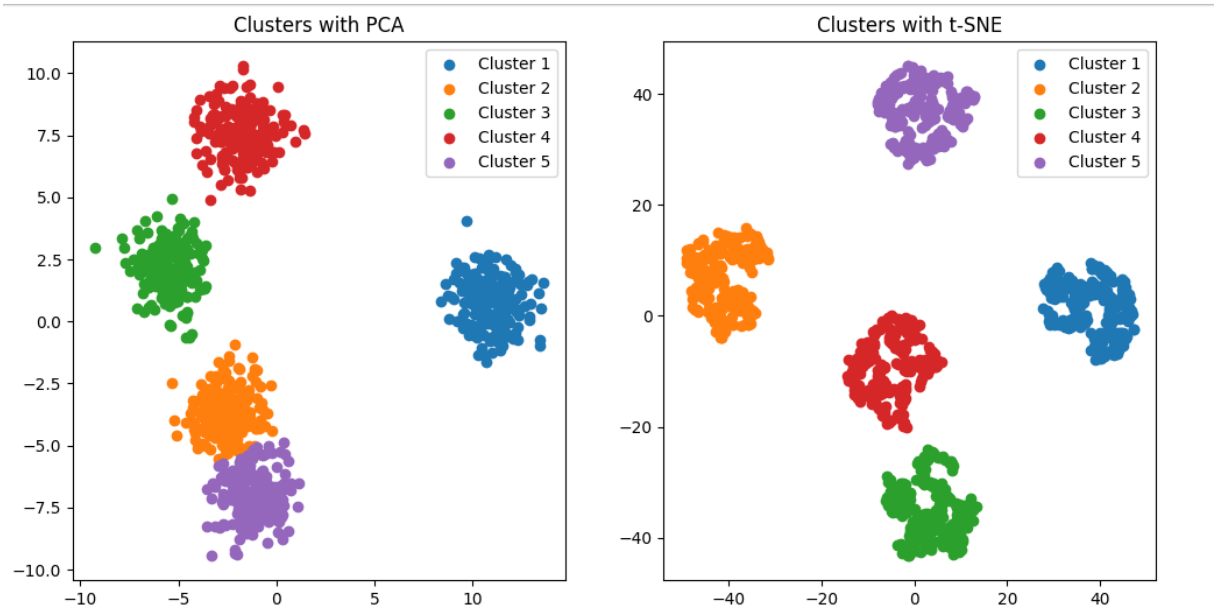
Partie 2 (Clustering):

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

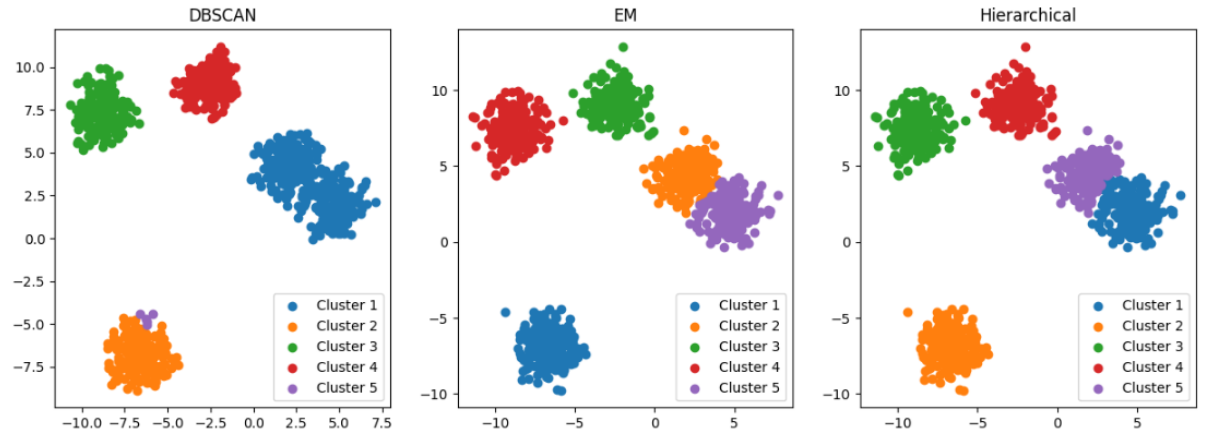




3. Présenter les clusters obtenues dans un graphe en utilisant matplotlib.



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


DBSCAN
EM
Hierarchical
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5