PROJET 3PDQ

Installation des packages

In [1]:
```python
# Packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
from sklearn.ensemble import IsolationForest
import warnings
# warnings.filterwarnings('ignore')
```

In [2]:
```python
dataset = pd.read_csv("3PDQ - Feuil1.csv")
dataset
```

Out[2]:
```
     N°de_Patient  Centre  Initiales  Sexe   Age  Date_Naissance  Date_visite1  Date_visite2  EVA_mean  EVA_max  ...  HADa.5  HADd.5  HADa.6  HADc
  0             1       1         BM     M  71.0         juil.-46    2019/05/22    2019/05/22      50.0     60.0  ...     0.0     0.0     0.0     0
  1             2       1         BM     M  49.0         sept.-69    2019/06/28    2019/06/28      35.0     80.0  ...     0.0     1.0     1.0     1
  2             3       1         NC     F  61.0         août-57    2019/07/05    2019/07/05      50.0     75.0  ...     2.0     2.0     3.0     1
  3             4       1         GE     F  65.0         juil.-53    2019/07/10    2019/07/10      54.0     80.0  ...     1.0     0.0     2.0     1
  4             5       1         ML     F  59.0         oct.-58    2019/07/17    2019/07/17      51.0     69.0  ...     1.0     2.0     3.0     2
...           ...     ...        ...   ...   ...              ...           ...           ...       ...      ...  ...     ...     ...     ...
220           221      14         GM     F  65.0         mai-55    2022/03/10    2022/03/11      70.0     80.0  ...     2.0     2.0     2.0     1
221           222      15         MM     M  76.0         févr.-45    2022/06/02    2022/06/02      80.0    100.0  ...     0.0     0.0     2.0     0
222           223      15         PA     F  64.0         nov.-56    2022/06/21    2022/06/21      45.0     70.0  ...     0.0     2.0     3.0     0
223           224      15         TL     M  60.0         févr.-61    2022/06/30    2022/06/30      40.0     70.0  ...     2.0     0.0     0.0     2
224           225      15         MC     F  45.0         nov.-76    2022/07/19    2022/07/19      88.0     90.0  ...     2.0     2.0     1.0     1

225 rows × 292 columns
```

In [3]:
```python
dataset.shape
```

Out[3]:
```
(225, 292)
```

In [4]:
```python
print(dataset.columns.tolist())
```

```
['N°de_Patient', 'Centre ', 'Initiales', 'Sexe', 'Age', 'Date_Naissance', 'Date_visite1', 'Date_visite2', 'EVA_mean', 'EVA_max', 'Inclusion1', 'Inclusion2', 'Inclusion
3', 'Inclusion4', 'Inclusion5', 'Inclusion6', 'Inclusion7', 'Non_Inclusion1', 'Non_Inclusion2', 'Non_Inclusion3', 'Non_Inclusion4', 'MOCA1', 'MOCA2', 'MOCA3', 'M
OCA4', 'MOCA5', 'MOCA6', 'MOCA7', 'MOCA8', 'MOCA9', 'MOCA10', 'MOCA_Total', 'item', 'item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8', 'item
9', 'item10', 'item11', 'item12', 'item13', 'item14', 'item15', 'item16', 'item17', 'item18', 'item19', 'item20', 'item21', 'item22', 'item23', 'item24', 'item25', 'item26', 'ite
m27', 'item28', 'item29', 'item30', 'item31', 'item32', 'item33', 'Douleur_Centrale', 'DN4.1', 'DN4.2', 'DN4.3', 'DN4.4', 'DN4.5', 'DN4.6', 'DN4.7', 'DN4.8', 'DN4.9', '
DN4.10', 'DN4.Total', 'KPPS1_s', 'KPPS1_f', 'Domain1_Total', 'KPPS2_s', 'KPPS2_f', 'KPPS3_s', 'KPPS3_f', 'Domain2_Total', 'KPPS4_s', 'KPPS4_f', 'KPPS5
_s', 'KPPS5_f', 'KPPS6_s', 'KPPS6_f', 'Domain3_Total', 'KPPS7_s', 'KPPS7_f', 'KPPS8_s', 'KPPS8_f', 'Domain4_Total', 'KPPS9_s', 'KPPS9_f', 'KPPS10_s', '
KPPS10_f', 'KPPS11_s', 'KPPS11_f', 'Domain5_Total', 'KPPS12_s', 'KPPS12_f', 'KPPS13_s', 'KPPS13_f', 'Domain6_Total', 'KPPS14_s', 'KPPS14_f', 'Domai
n7_Total', 'KPPS_Total_Score', 'MDS_UPDRS1.A', 'MDS_UPDRS1.1', 'MDS_UPDRS1.2', 'MDS_UPDRS1.3', 'MDS_UPDRS1.4', 'MDS_UPDRS1.5', 'MDS_U
PDRS1.6', 'MDS_UPDRS1.6a', 'MDS_UPDRS1.7', 'MDS_UPDRS1.8', 'MDS_UPDRS1.9', 'MDS_UPDRS1.10', 'MDS_UPDRS1.11', 'MDS_UPDRS1.12', 'MDS
_UPDRS1.13', 'P1_MDS_UPDRS', 'MDS_UPDRS2.1', 'MDS_UPDRS2.2', 'MDS_UPDRS2.3', 'MDS_UPDRS2.4', 'MDS_UPDRS2.5', 'MDS_UPDRS2.6', 'MDS
_UPDRS2.7', 'MDS_UPDRS2.8', 'MDS_UPDRS2.9', 'MDS_UPDRS2.10', 'MDS_UPDRS2.11', 'MDS_UPDRS2.12', 'MDS_UPDRS2.13', 'P2_MDS_UPDRS', '
MDS_UPDRS3a', 'MDS_UPDRS3b', 'MDS_UPDRS3c', 'MDS_UPDRS3.C1', 'MDS_UPDRS3.1', 'MDS_UPDRS3.2', 'MDS_UPDRS3.3a', 'MDS_UPDRS3.3b', '
MDS_UPDRS3.3c', 'MDS_UPDRS3.3d', 'MDS_UPDRS3.3e', 'MDS_UPDRS3.4a', 'MDS_UPDRS3.4b', 'MDS_UPDRS3.5a', 'MDS_UPDRS3.5b', 'MDS_UPDR
S3.6a', 'MDS_UPDRS3.6b', 'MDS_UPDRS3.7a', 'MDS_UPDRS3.7b', 'MDS_UPDRS3.8a', 'MDS_UPDRS3.8b', 'MDS_UPDRS3.9', 'MDS_UPDRS3.10', 'MDS
_UPDRS3.11', 'MDS_UPDRS3.12', 'MDS_UPDRS3.13', 'MDS_UPDRS3.14', 'MDS_UPDRS3.15a', 'MDS_UPDRS3.15b', 'MDS_UPDRS3.16a', 'MDS_UPDR
S3.16b', 'MDS_UPDRS3.17a', 'MDS_UPDRS3.17b', 'MDS_UPDRS3.17c', 'MDS_UPDRS3.17d', 'MDS_UPDRS3.17e', 'MDS_UPDRS3.18', 'P3_MDS_UPDR
S', 'MDS_UPDRSdiskiné', 'MDS_UPDRSinterféré', 'MDS_UPDRS_Hoehn&Yahr', 'MDS_UPDRS4.1', 'MDS_UPDRS4.2', 'MDS_UPDRS4.3', 'MDS_UPDRS4.4
', 'MDS_UPDRS4.5', 'MDS_UPDRS4.6', 'P4_MDS_UPDRS', 'MDS_UPDS_Total', 'McGilla1', 'McGilla2', 'McGilla3', 'McGilla4', 'McGilla5', 'McGilla6', 'McGilla7'
, 'McGilla8', 'McGilla9', 'McGilla10', 'McGilla11', 'McGilla12', 'McGilla13', 'McGilla14', 'McGilla15', 'McGillb', 'McGill_Total', 'BPIa', 'BPIb', 'BPIc', 'BPId', 'BPIe', 'B
PIf', 'BPIg', 'BPI_Total', 'PCS1', 'PCS2', 'PCS3', 'PCS4', 'PCS5', 'PCS6', 'PCS7', 'PCS8', 'PCS9', 'PCS10', 'PCS11', 'PCS12', 'PCS13', 'PCS_Total', 'PDQ1', 'P
DQ2', 'PDQ3', 'PDQ4', 'PDQ5', 'PDQ6', 'PDQ7', 'PDQ8', 'PDQ9', 'PDQ10', 'PDQ11', 'PDQ12', 'PDQ13', 'PDQ14', 'PDQ15', 'PDQ16', 'PDQ17', 'PDQ18', 'PDQ19
', 'PDQ20', 'PDQ21', 'PDQ22', 'PDQ23', 'PDQ24', 'PDQ25', 'PDQ26', 'PDQ27', 'PDQ28', 'PDQ29', 'PDQ30', 'PDQ31', 'PDQ32', 'PDQ33', 'PDQ34', 'PDQ35', 'P
DQ36', 'PDQ37', 'PDQ38', 'PDQ39', 'PDQ_Total', 'PDQ_Score_Complète', ' PDQ_Résultat_Pourcentage', 'HADa.1', 'HADd.1', 'HADa.2', 'HADd.2', 'HADa.3', '
HADd.3', 'HADa.4', 'HADd.4', 'HADa.5', 'HADd.5', 'HADa.6', 'HADd.6', 'HADa.7', 'HADd.7', 'Anxiété_Total', 'Depression_Total', 'Total_Score', 'Conformité_au_
protocol']
```

In [5]:
```python
df= dataset[['item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8', 'item9', 'item10', 'item11', 'item12', 'item13', 'item14', 'item15', 'item16', 'item17
'item19', 'item20', 'item21', 'item22', 'item23', 'item24', 'item25', 'item26', 'item27', 'item28','Douleur_Centrale']]
df
```

```
Out[5]:
      item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 ... item20 item21 item22 item23 item24 item25 item26 item27 item28
   0    0.0   1.0     1   0.0     1     0   1.0     0   0.0      0 ...      0      0      1      1      0      1      0      0      0
   1    1.0   0.0     0   1.0     0     0   1.0     1   0.0      0 ...      0      0      1      1      1      1      1      0      0
   2    0.0   0.0     0   0.0     1     0   1.0     0   0.0      0 ...      0      0      0      0      0      0      1      0      0
   3    0.0   1.0     1   0.0     0     0   1.0     0   1.0      0 ...      0      0      0      1      1      0      0      0      0
   4    1.0   0.0     1   1.0     1     0   0.0     1   0.0      0 ...      0      1      0      1      1      0      0      0      0
 ...    ...   ...   ...   ...   ...   ...   ...   ...   ...    ... ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
 220    0.0   1.0     1   0.0     0     0   1.0     1   0.0      0 ...      0      0      1      1      1      1      1      0      0
 221    1.0   1.0     1   0.0     1     0   1.0     0   0.0      0 ...      0      0      0      1      1      1      0      1      0
 222    0.0   0.0     0   0.0     0     0   0.0     0   1.0      1 ...      0      0      0      1      1      0      0      0      0
 223    0.0   0.0     0   0.0     1     0   0.0     1   0.0      1 ...      0      0      1      1      1      1      1      1      0
 224    1.0   0.0     0   0.0     1     1   1.0     0   0.0      0 ...      1      1      1      1      1      1      1      1      1

225 rows × 29 columns
```

```
In [6]: df.shape
```

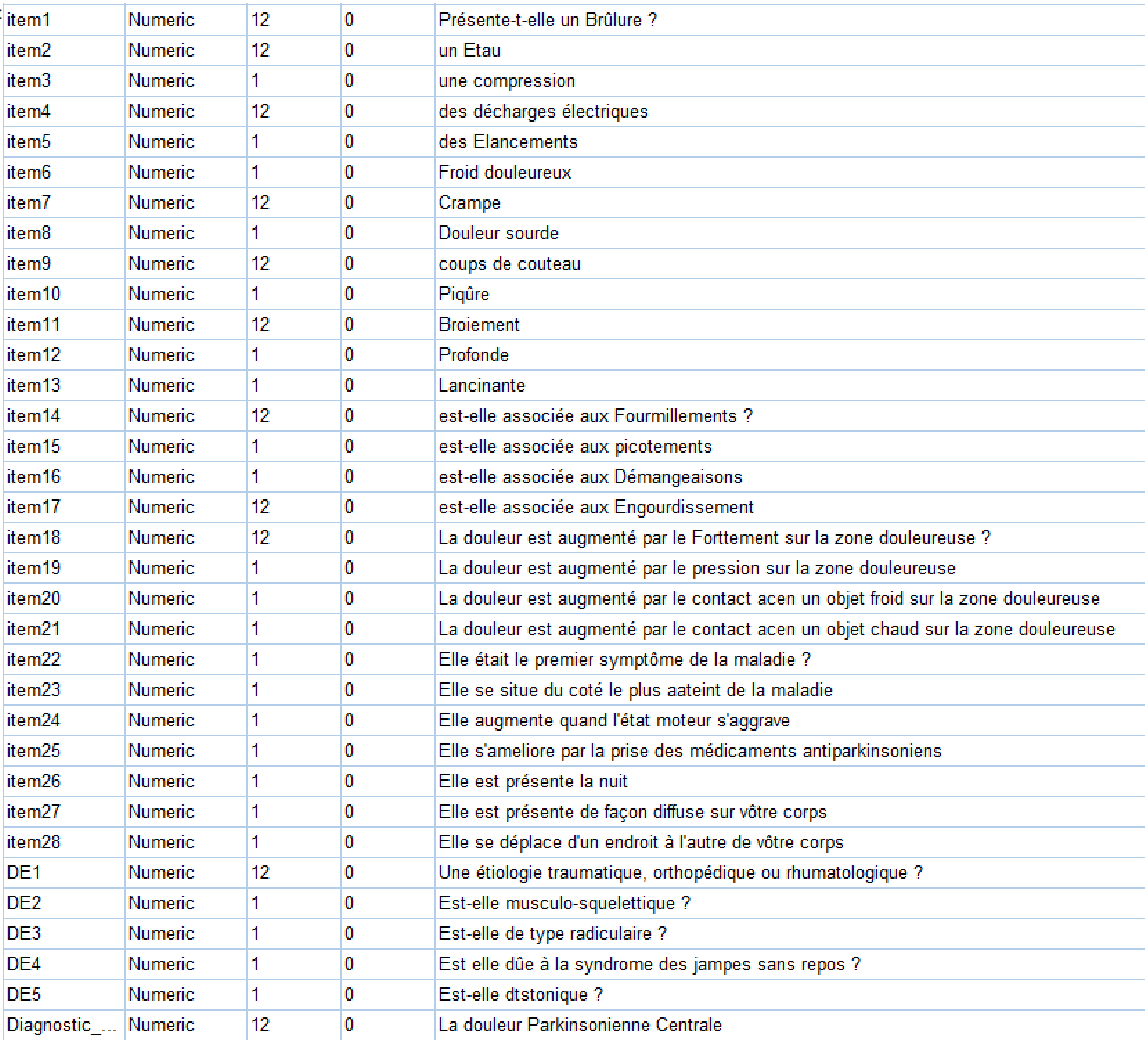
```
Out[6]:(225, 29)
```

Objectif :

1- Prédire la variable Y qui est la douleur centrale
2- Sortir les items qui expliquent le mieux mon Y

```
In [7]: from IPython.display import Image

        # Afficher l'image
        Image(filename="C:/Users/aliah/Downloads/Capture d'écran 2023-04-21 140544.png")
```

Out[7]:



item1	Numeric	12	0	Présente-t-elle un Brûlure ?
item2	Numeric	12	0	un Etau
item3	Numeric	1	0	une compression
item4	Numeric	12	0	des décharges électriques
item5	Numeric	1	0	des Elancements
item6	Numeric	1	0	Froid douleureux
item7	Numeric	12	0	Crampe
item8	Numeric	1	0	Douleur sourde
item9	Numeric	12	0	coups de couteau
item10	Numeric	1	0	Piqûre
item11	Numeric	12	0	Broiement
item12	Numeric	1	0	Profonde
item13	Numeric	1	0	Lancinante
item14	Numeric	12	0	est-elle associée aux Fourmillements ?
item15	Numeric	1	0	est-elle associée aux picotements
item16	Numeric	1	0	est-elle associée aux Démangeaisons
item17	Numeric	12	0	est-elle associée aux Engourdissement
item18	Numeric	12	0	La douleur est augmenté par le Forttement sur la zone douleureuse ?
item19	Numeric	1	0	La douleur est augmenté par le pression sur la zone douleureuse
item20	Numeric	1	0	La douleur est augmenté par le contact acen un objet froid sur la zone douleureuse
item21	Numeric	1	0	La douleur est augmenté par le contact acen un objet chaud sur la zone douleureuse
item22	Numeric	1	0	Elle était le premier symptôme de la maladie ?
item23	Numeric	1	0	Elle se situe du coté le plus aateint de la maladie
item24	Numeric	1	0	Elle augmente quand l'état moteur s'aggrave
item25	Numeric	1	0	Elle s'ameliore par la prise des médicaments antiparkinsoniens
item26	Numeric	1	0	Elle est présente la nuit
item27	Numeric	1	0	Elle est présente de façon diffuse sur vôtre corps
item28	Numeric	1	0	Elle se déplace d'un endroit à l'autre de vôtre corps
DE1	Numeric	12	0	Une étiologie traumatique, orthopédique ou rhumatologique ?
DE2	Numeric	1	0	Est-elle musculo-squelettique ?
DE3	Numeric	1	0	Est-elle de type radiculaire ?
DE4	Numeric	1	0	Est elle dûe à la syndrome des jampes sans repos ?
DE5	Numeric	1	0	Est-elle dtstonique ?
Diagnostic_...	Numeric	12	0	La douleur Parkinsonienne Centrale

Vérifier le type de chaque variable

```
In [8]: df.select_dtypes(object).columns
```

```
Out[8]:Index(['item3', 'item5', 'item6', 'item8', 'item10', 'item12', 'item13',
              'item15', 'item16', 'item19', 'item20', 'item21', 'item22', 'item23',
              'item24', 'item25', 'item26', 'item27', 'item28'],
             dtype='object')
```

Conversion de toutes les variables en float

```
In [9]: # Convertir les colonnes object en float
        for col in df.columns:
            if df[col].dtype == 'object':
                df[col] = pd.to_numeric(df[col], errors='coerce')

        # Afficher les types de données des colonnes
        print(df.dtypes)
```

```
C:\Users\aliah\AppData\Local\Temp\ipykernel_20984\1561378193.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = pd.to_numeric(df[col], errors='coerce')
item1              float64
item2              float64
item3              float64
item4              float64
item5              float64
item6              float64
item7              float64
item8              float64
item9              float64
item10             float64
item11             float64
item12             float64
item13             float64
item14             float64
item15             float64
item16             float64
item17             float64
item18             float64
item19             float64
item20             float64
item21             float64
item22             float64
item23             float64
item24             float64
item25             float64
item26             float64
item27             float64
item28             float64
Douleur_Centrale    float64
dtype: object
```

Gestion des données manquantes

```
In [10]: #Compter le nombre de valeurs manquantes par variable
         missing_values_count = df.isnull().sum()

         # Afficher le nombre de valeurs manquantes par variable
         print("Nombre de valeurs manquantes par variable :\n", missing_values_count)

         # Afficher les variables qui contiennent des valeurs manquantes
         missing_variables = df.columns[df.isnull().any()].tolist()
         print("\nVariables avec des valeurs manquantes :", missing_variables)
```

```
Nombre de valeurs manquantes par variable :
 item1               8
item2               8
item3               8
item4               8
item5              10
item6              10
item7               7
item8               8
item9               8
item10              9
item11              8
item12             10
item13              9
item14              6
item15              7
item16              7
item17              6
item18              7
item19              8
item20              7
item21              8
item22              8
item23              9
item24             11
item25              8
item26              8
item27              9
item28              7
Douleur_Centrale     5
dtype: int64

Variables avec des valeurs manquantes : ['item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8', 'item9', 'item10', 'item11', 'item12', 'item13', 'item14', 'it
em15', 'item16', 'item17', 'item18', 'item19', 'item20', 'item21', 'item22', 'item23', 'item24', 'item25', 'item26', 'item27', 'item28', 'Douleur_Centrale']
```

In [11]:
```
df.isnull().sum().sum()
```

```
Out[11]:232
```

Interpretation pour savoir la manière manière de gérer les nan

Cependant, il est important de noter que l'imputation par la moyenne peut ne pas être la méthode la plus appropriée pour gérer les données manquantes dans un jeu de données, en particulier pour les variables binaires. Il existe d'autres méthodes d'imputation telles que la médiane, le mode ou la méthode KNN qui peuvent mieux convenir à ce type de données.

IMPUTATION par la méthode KNN (k-Nearest Neighbors)

La méthode KNN (k-Nearest Neighbors) est une méthode d'imputation de données manquantes basée sur les données existantes. Elle consiste à remplacer chaque valeur manquante dans une variable par la valeur de la variable la plus proche, en termes de distance euclidienne, des k échantillons les plus proches de la variable contenant la valeur manquante.

In [12]:
```python
from sklearn.impute import KNNImputer

# Imputer les valeurs manquantes avec KNN
imputer = KNNImputer(n_neighbors=5)
imputed_data = imputer.fit_transform(df)

# Convertir les données imputées en DataFrame
df_imputed = pd.DataFrame(imputed_data, columns=df.columns)

# Renommer les colonnes
column_names = list(df.columns)
for i in range(len(column_names)):
    df_imputed.rename(columns={i: column_names[i]}, inplace=True)

# Afficher le DataFrame imputé avec les noms de colonnes corrects
print(df_imputed)
```

```
     item1  item2  item3  item4  item5  item6  item7  item8  item9  item10  \
0      0.0    1.0    1.0    0.0    1.0    0.0    1.0    0.0    0.0     0.0
1      1.0    0.0    0.0    1.0    0.0    0.0    1.0    1.0    0.0     0.0
2      0.0    0.0    0.0    0.0    1.0    0.0    1.0    0.0    0.0     0.0
3      0.0    1.0    1.0    0.0    0.0    0.0    1.0    0.0    1.0     0.0
4      1.0    0.0    1.0    1.0    1.0    0.0    0.0    1.0    0.0     0.0
..     ...    ...    ...    ...    ...    ...    ...    ...    ...     ...
220    0.0    1.0    1.0    0.0    0.0    0.0    1.0    1.0    0.0     0.0
221    1.0    1.0    1.0    0.0    1.0    0.0    1.0    0.0    0.0     0.0
222    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    1.0     1.0
223    0.0    0.0    0.0    0.0    1.0    0.0    0.0    1.0    0.0     1.0
224    1.0    0.0    0.0    0.0    1.0    1.0    1.0    0.0    0.0     0.0

     ...  item20  item21  item22  item23  item24  item25  item26  item27  \
0    ...     0.0     0.0     1.0     1.0     0.0     1.0     0.0     0.0
1    ...     0.0     0.0     1.0     1.0     1.0     1.0     1.0     0.0
2    ...     0.0     0.0     0.0     0.0     0.0     0.0     1.0     0.0
3    ...     0.0     0.0     0.0     1.0     1.0     0.0     0.0     0.0
4    ...     0.0     1.0     0.0     1.0     1.0     0.0     0.0     0.0
..   ...     ...     ...     ...     ...     ...     ...     ...     ...
220  ...     0.0     0.0     1.0     1.0     1.0     1.0     1.0     0.0
221  ...     0.0     0.0     0.0     1.0     1.0     1.0     0.0     1.0
222  ...     0.0     0.0     0.0     1.0     1.0     0.0     0.0     0.0
223  ...     0.0     0.0     1.0     1.0     1.0     1.0     1.0     1.0
224  ...     1.0     1.0     1.0     1.0     1.0     1.0     1.0     1.0

     item28  Douleur_Centrale
0       0.0               1.0
1       0.0               1.0
2       0.0               0.0
3       0.0               0.0
4       0.0               1.0
..      ...               ...
220     0.0               0.0
221     0.0               1.0
222     0.0               1.0
223     0.0               1.0
224     1.0               1.0

[225 rows x 29 columns]
```

Commentaire :

Le paramètre n_neighbors égal à 5, qui spécifie le nombre de voisins à considérer lors de l'imputation

In [13]:

```python
#Compter le nombre de valeurs manquantes par variable
missing_values_count = df_imputed.isnull().sum()

# Afficher le nombre de valeurs manquantes par variable
print("Nombre de valeurs manquantes par variable :\n", missing_values_count)

# Afficher les variables qui contiennent des valeurs manquantes
missing_variables = df_imputed.columns[df.isnull().any()].tolist()
print("\nVariables avec des valeurs manquantes :", missing_variables)
```

```
Nombre de valeurs manquantes par variable :
 item1              0
item2              0
item3              0
item4              0
item5              0
item6              0
item7              0
item8              0
item9              0
item10             0
item11             0
item12             0
item13             0
item14             0
item15             0
item16             0
item17             0
item18             0
item19             0
item20             0
item21             0
item22             0
item23             0
item24             0
item25             0
item26             0
item27             0
item28             0
Douleur_Centrale    0
dtype: int64

Variables avec des valeurs manquantes : ['item1', 'item2', 'item3', 'item4', 'item5', 'item6', 'item7', 'item8', 'item9', 'item10', 'item11', 'item12', 'item13', 'item14', 'it
em15', 'item16', 'item17', 'item18', 'item19', 'item20', 'item21', 'item22', 'item23', 'item24', 'item25', 'item26', 'item27', 'item28', 'Douleur_Centrale']
```

In [14]:
```python
df_imputed.isnull().sum().sum()
```

Out[14]:0

In [15]:
```python
df_imputed
```

Out[15]:

	item1	item2	item3	item4	item5	item6	item7	item8	item9	item10	...	item20	item21	item22	item23	item24	item25	item26	item27	item2
0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0
1	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	...	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0
2	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0
3	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0
4	1.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
220	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	...	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0
221	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0
222	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	...	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0
223	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0
224	1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1

225 rows × 29 columns

Voir comment mes données sont distribuées

Les variables binaires ne peuvent pas être analysées avec les mêmes méthodes que les variables continues. En effet, les variables binaires ne peuvent prendre que deux valeurs possibles (0 ou 1), ce qui signifie que la distribution ne peut pas être représentée par une courbe de densité, un histogramme ou un diagramme en boîte.

Cependant, il est toujours important de comprendre comment les valeurs sont réparties dans votre ensemble de données. Pour les variables binaires, vous pouvez calculer la proportion de 0 et de 1 dans votre jeu de données, ainsi que la fréquence de chaque catégorie. Vous pouvez également utiliser des tableaux de contingence pour voir comment les variables binaires sont associées les unes aux autres.

En résumé, la meilleure façon de comprendre comment les variables binaires sont distribuées est de calculer les proportions et les fréquences, et de les comparer à d'autres variables binaires pour voir comment elles sont associées.

In [16]:
```python
# Voir la distribution pour chaque variable
for col in df_imputed.columns:
    if df_imputed[col].dtype == 'float64':
        df_imputed[col] = df_imputed[col].astype(int)
    zeros = np.count_nonzero(df_imputed[col] == 0)
    ones = np.count_nonzero(df_imputed[col] == 1)
    print(f"Variable {col}:")
```

```
        print(f"Nombre de 0 : {zeros}")
        print(f"Nombre de 1 : {ones}")
        print(f"Proportion de 0 : {zeros/len(df_imputed[col])}")
        print(f"Proportion de 1 : {ones/len(df_imputed[col])}")
        print(f"Fréquence de 0 : {zeros/(zeros+ones)}")
        print(f"Fréquence de 1 : {ones/(zeros+ones)}")
        print("")
```

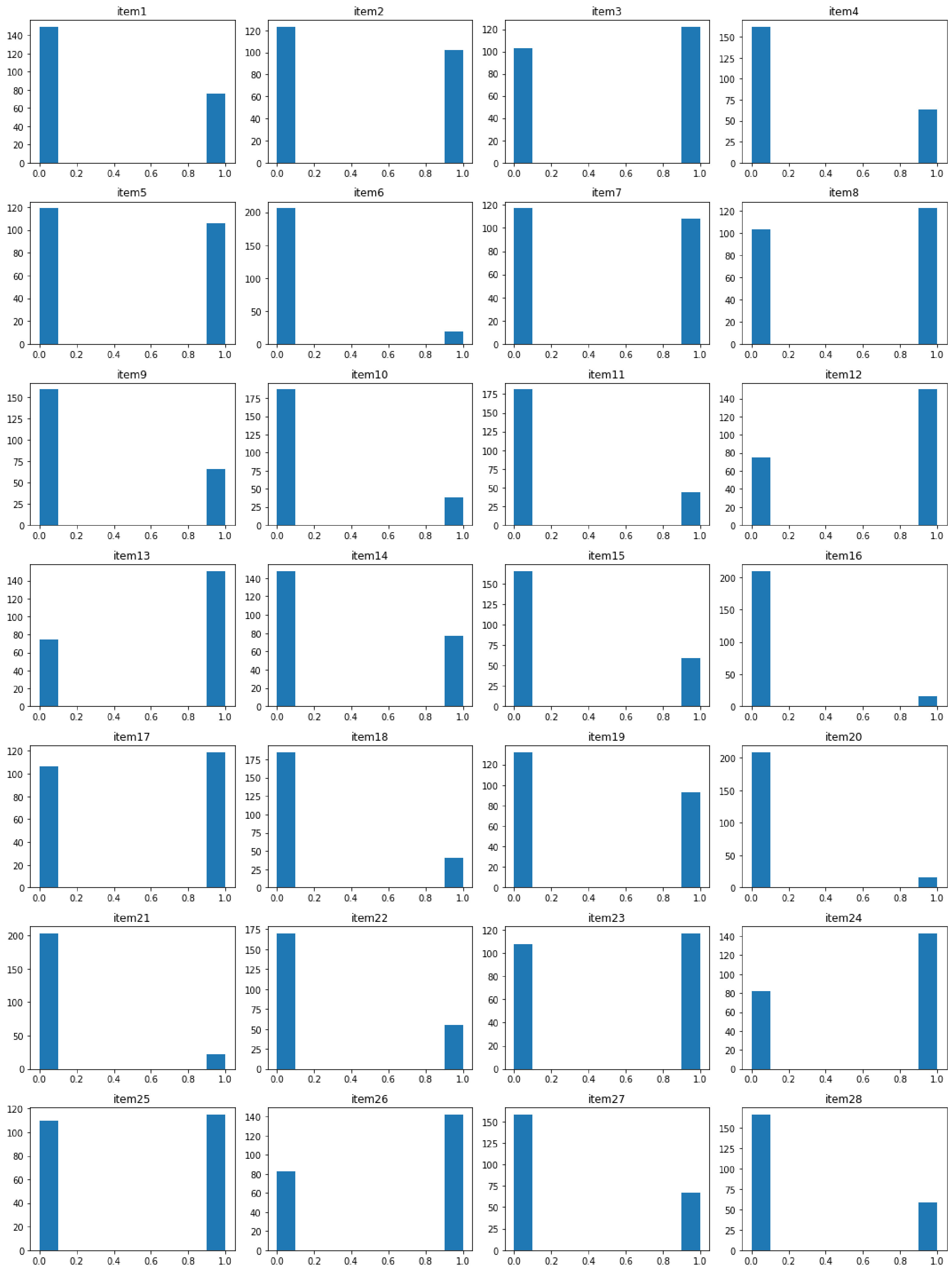
```
Variable item1:
Nombre de 0 : 149
Nombre de 1 : 76
Proportion de 0 : 0.6622222222222223
Proportion de 1 : 0.3377777777777778
Fréquence de 0 : 0.6622222222222223
Fréquence de 1 : 0.3377777777777778

Variable item2:
Nombre de 0 : 123
Nombre de 1 : 102
Proportion de 0 : 0.5466666666666666
Proportion de 1 : 0.4533333333333333
Fréquence de 0 : 0.5466666666666666
Fréquence de 1 : 0.4533333333333333

Variable item3:
Nombre de 0 : 103
Nombre de 1 : 122
Proportion de 0 : 0.4577777777777778
Proportion de 1 : 0.5422222222222223
Fréquence de 0 : 0.4577777777777778
Fréquence de 1 : 0.5422222222222223

Variable item4:
Nombre de 0 : 162
Nombre de 1 : 63
Proportion de 0 : 0.72
Proportion de 1 : 0.28
Fréquence de 0 : 0.72
Fréquence de 1 : 0.28

Variable item5:
Nombre de 0 : 119
Nombre de 1 : 106
Proportion de 0 : 0.5288888888888889
Proportion de 1 : 0.4711111111111111
Fréquence de 0 : 0.5288888888888889
Fréquence de 1 : 0.4711111111111111

Variable item6:
Nombre de 0 : 206
Nombre de 1 : 19
Proportion de 0 : 0.9155555555555556
Proportion de 1 : 0.08444444444444445
Fréquence de 0 : 0.9155555555555556
Fréquence de 1 : 0.08444444444444445

Variable item7:
Nombre de 0 : 117
Nombre de 1 : 108
Proportion de 0 : 0.52
Proportion de 1 : 0.48
Fréquence de 0 : 0.52
Fréquence de 1 : 0.48

Variable item8:
Nombre de 0 : 103
Nombre de 1 : 122
Proportion de 0 : 0.4577777777777778
Proportion de 1 : 0.5422222222222223
Fréquence de 0 : 0.4577777777777778
Fréquence de 1 : 0.5422222222222223

Variable item9:
Nombre de 0 : 159
Nombre de 1 : 66
Proportion de 0 : 0.7066666666666667
Proportion de 1 : 0.29333333333333333
Fréquence de 0 : 0.7066666666666667
Fréquence de 1 : 0.29333333333333333

Variable item10:
Nombre de 0 : 187
Nombre de 1 : 38
```

```
Proportion de 0 : 0.8311111111111111
Proportion de 1 : 0.1688888888888889
Fréquence de 0 : 0.8311111111111111
Fréquence de 1 : 0.1688888888888889

Variable item11:
Nombre de 0 : 181
Nombre de 1 : 44
Proportion de 0 : 0.8044444444444444
Proportion de 1 : 0.19555555555555557
Fréquence de 0 : 0.8044444444444444
Fréquence de 1 : 0.19555555555555557

Variable item12:
Nombre de 0 : 75
Nombre de 1 : 150
Proportion de 0 : 0.3333333333333333
Proportion de 1 : 0.6666666666666666
Fréquence de 0 : 0.3333333333333333
Fréquence de 1 : 0.6666666666666666

Variable item13:
Nombre de 0 : 74
Nombre de 1 : 151
Proportion de 0 : 0.3288888888888889
Proportion de 1 : 0.6711111111111111
Fréquence de 0 : 0.3288888888888889
Fréquence de 1 : 0.6711111111111111

Variable item14:
Nombre de 0 : 148
Nombre de 1 : 77
Proportion de 0 : 0.6577777777777778
Proportion de 1 : 0.3422222222222222
Fréquence de 0 : 0.6577777777777778
Fréquence de 1 : 0.3422222222222222

Variable item15:
Nombre de 0 : 166
Nombre de 1 : 59
Proportion de 0 : 0.7377777777777778
Proportion de 1 : 0.26222222222222225
Fréquence de 0 : 0.7377777777777778
Fréquence de 1 : 0.26222222222222225

Variable item16:
Nombre de 0 : 210
Nombre de 1 : 15
Proportion de 0 : 0.9333333333333333
Proportion de 1 : 0.06666666666666667
Fréquence de 0 : 0.9333333333333333
Fréquence de 1 : 0.06666666666666667

Variable item17:
Nombre de 0 : 106
Nombre de 1 : 119
Proportion de 0 : 0.4711111111111111
Proportion de 1 : 0.5288888888888889
Fréquence de 0 : 0.4711111111111111
Fréquence de 1 : 0.5288888888888889

Variable item18:
Nombre de 0 : 185
Nombre de 1 : 40
Proportion de 0 : 0.8222222222222222
Proportion de 1 : 0.17777777777777778
Fréquence de 0 : 0.8222222222222222
Fréquence de 1 : 0.17777777777777778

Variable item19:
Nombre de 0 : 132
Nombre de 1 : 93
Proportion de 0 : 0.5866666666666667
Proportion de 1 : 0.41333333333333333
Fréquence de 0 : 0.5866666666666667
Fréquence de 1 : 0.41333333333333333

Variable item20:
Nombre de 0 : 209
Nombre de 1 : 16
Proportion de 0 : 0.9288888888888889
Proportion de 1 : 0.07111111111111111
Fréquence de 0 : 0.9288888888888889
```

```
Fréquence de 0 : 0.9288888888888889
Fréquence de 1 : 0.07111111111111111

Variable item21:
Nombre de 0 : 203
Nombre de 1 : 22
Proportion de 0 : 0.9022222222222223
Proportion de 1 : 0.09777777777777778
Fréquence de 0 : 0.9022222222222223
Fréquence de 1 : 0.09777777777777778

Variable item22:
Nombre de 0 : 170
Nombre de 1 : 55
Proportion de 0 : 0.7555555555555555
Proportion de 1 : 0.24444444444444444
Fréquence de 0 : 0.7555555555555555
Fréquence de 1 : 0.24444444444444444

Variable item23:
Nombre de 0 : 108
Nombre de 1 : 117
Proportion de 0 : 0.48
Proportion de 1 : 0.52
Fréquence de 0 : 0.48
Fréquence de 1 : 0.52

Variable item24:
Nombre de 0 : 82
Nombre de 1 : 143
Proportion de 0 : 0.36444444444444446
Proportion de 1 : 0.6355555555555555
Fréquence de 0 : 0.36444444444444446
Fréquence de 1 : 0.6355555555555555

Variable item25:
Nombre de 0 : 110
Nombre de 1 : 115
Proportion de 0 : 0.4888888888888889
Proportion de 1 : 0.5111111111111111
Fréquence de 0 : 0.4888888888888889
Fréquence de 1 : 0.5111111111111111

Variable item26:
Nombre de 0 : 83
Nombre de 1 : 142
Proportion de 0 : 0.3688888888888889
Proportion de 1 : 0.6311111111111111
Fréquence de 0 : 0.3688888888888889
Fréquence de 1 : 0.6311111111111111

Variable item27:
Nombre de 0 : 158
Nombre de 1 : 67
Proportion de 0 : 0.7022222222222222
Proportion de 1 : 0.29777777777777775
Fréquence de 0 : 0.7022222222222222
Fréquence de 1 : 0.29777777777777775

Variable item28:
Nombre de 0 : 166
Nombre de 1 : 59
Proportion de 0 : 0.7377777777777778
Proportion de 1 : 0.26222222222222225
Fréquence de 0 : 0.7377777777777778
Fréquence de 1 : 0.26222222222222225

Variable Douleur_Centrale:
Nombre de 0 : 144
Nombre de 1 : 81
Proportion de 0 : 0.64
Proportion de 1 : 0.36
Fréquence de 0 : 0.64
Fréquence de 1 : 0.36
```

In [17]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(nrows=7, ncols=4, figsize=(15, 20))
for i, ax in enumerate(axes.flatten()):
    if i < len(df_imputed.columns):
        col = df_imputed.columns[i]
        ax.hist(df_imputed[col])
        ax.set_title(col)
    plt.tight_layout()
    plt.show()
```



MODELISATION

In [18]:
```
# 1) Créer une matrice des variables indépendantes et le vecteur de la variable dépendante.
# X est la matrice et Y est le vecteur
# La matrice des variables indépendantes est aussi appeelée matrice de featuresµ

X = df_imputed.drop('Douleur_Centrale', axis=1)  # Supprimer la colonne "target" de la matrice X
Y = df_imputed['Douleur_Centrale']             # Sélectionner uniquement la colonne "target" pour Y
```

In [19]:
```
X
```

Out[19]:

	item1	item2	item3	item4	item5	item6	item7	item8	item9	item10	...	item19	item20	item21	item22	item23	item24	item25	item26	item2
0	0	1	1	0	1	0	1	0	0	0	...	0	0	0	1	1	0	1	0	
1	1	0	0	1	0	0	1	1	0	0	...	0	0	0	1	1	1	1	1	
2	0	0	0	0	1	0	1	0	0	0	...	1	0	0	0	0	0	0	1	
3	0	1	1	0	0	0	1	0	1	0	...	1	0	0	0	1	1	0	0	
4	1	0	1	1	1	0	0	1	0	0	...	1	0	1	0	1	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
220	0	1	1	0	0	0	1	1	0	0	...	1	0	0	1	1	1	1	1	
221	1	1	1	0	1	0	1	0	0	0	...	0	0	0	0	1	1	1	0	
222	0	0	0	0	0	0	0	0	1	1	...	1	0	0	0	1	1	0	0	
223	0	0	0	0	1	0	0	1	0	1	...	0	0	0	1	1	1	1	1	
224	1	0	0	0	1	1	1	0	0	0	...	1	1	1	1	1	1	1	1	

225 rows × 28 columns

In [20]:
```
Y
```

Out[20]:
```
0      1
1      1
2      0
3      0
4      1
      ..
220    0
221    1
222    1
223    1
224    1
Name: Douleur_Centrale, Length: 225, dtype: int32
```

Rééquilibré mon jeu de données par la méthode SMOT

La sur-échantillonnage de la classe minoritaire (oversampling): cela consiste à ajouter des copies d'observations de la classe minoritaire pour atteindre un équilibre avec la classe majoritaire. Cette technique peut être réalisée à l'aide de méthodes telles que la duplication aléatoire ou la génération synthétique de données (SMOTE)

SMOTE (Synthetic Minority Over-sampling Technique) est une méthode d'oversampling qui consiste à créer des exemples synthétiques de la classe minoritaire en interpolant de manière aléatoire les échantillons existants. Elle permet d'augmenter la taille de la classe minoritaire sans générer de copies exactes des échantillons existants, ce qui peut améliorer la généralisation du modèle

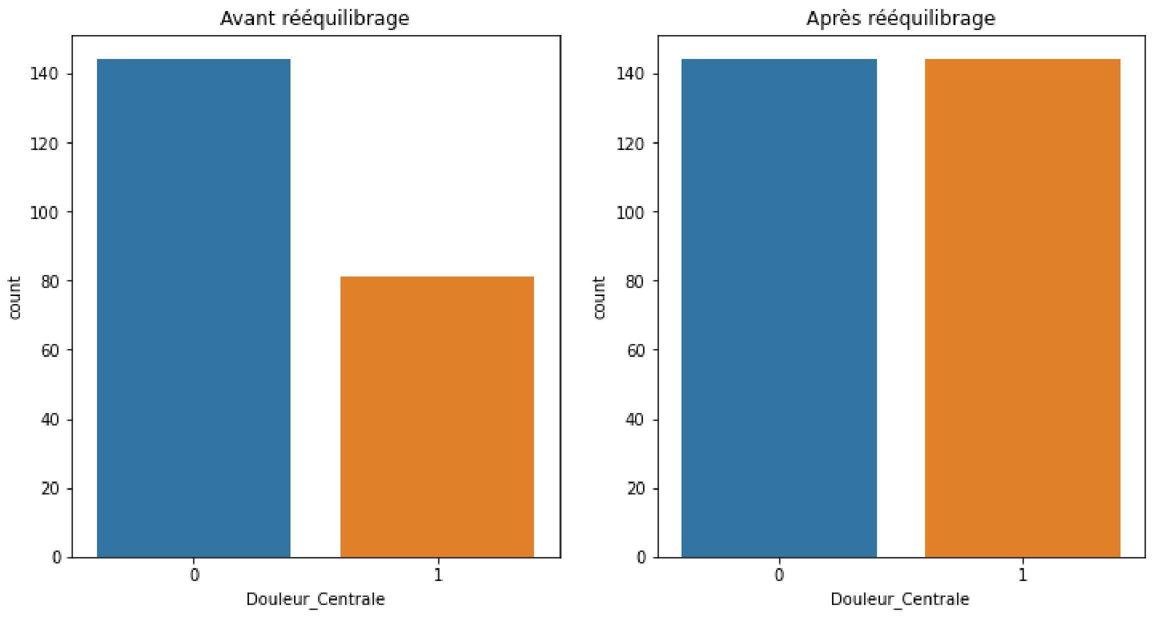
In [21]:
```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from imbalanced-learn) (1.21.5)
Requirement already satisfied: joblib>=1.1.1 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from imbalanced-learn) (1.1.1)
Requirement already satisfied: scipy>=1.3.2 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from imbalanced-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from imbalanced-learn) (2.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from imbalanced-learn) (1.0.2)
```

In [22]:
```
from imblearn.over_sampling import SMOTE

# Créer un objet SMOTE
smote = SMOTE()

# Appliquer SMOTE sur les données
X_resampled, Y_resampled = smote.fit_resample(X, Y)
```

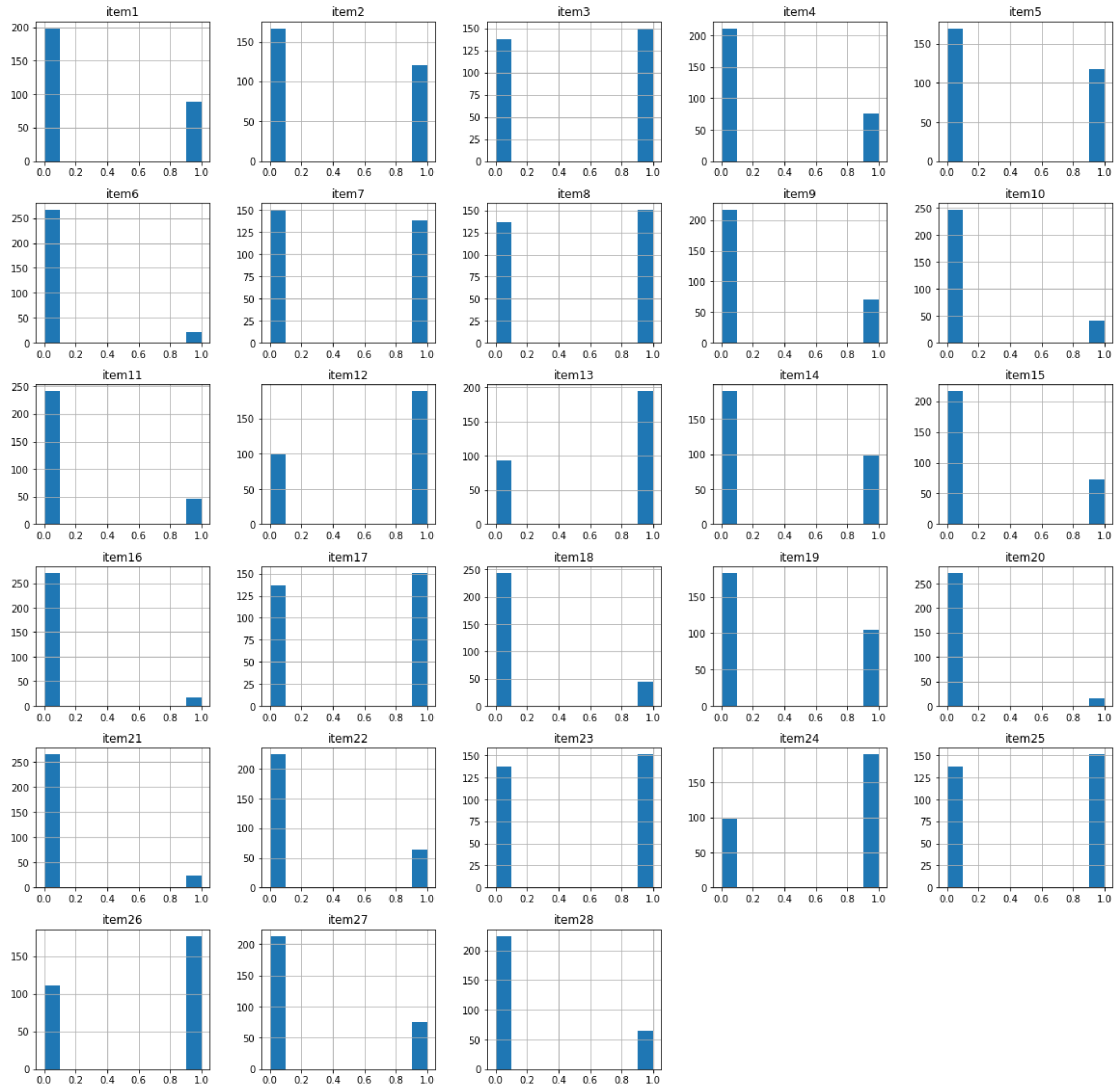
In [23]:
```
# Vérifier la distribution des classes avant et après rééquilibrage
fig, axs = plt.subplots(ncols=2, figsize=(12,6))
sns.countplot(Y, ax=axs[0])
sns.countplot(Y_resampled, ax=axs[1])
axs[0].set_title("Avant rééquilibrage")
axs[1].set_title("Après rééquilibrage")
plt.show()
```

```
C:\Users\aliah\anaconda3\envs\PythonProject\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
C:\Users\aliah\anaconda3\envs\PythonProject\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```



In [24]:

```
# Afficher la distribution des variables de X après rééquilibrage
X_resampled.hist(figsize=(20,20))
plt.show()
```



Séparation du dataset en training_set et en test_set

In [25]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X_resampled, Y_resampled, test_size = 1/3, random_state= 0)

# Calculer la proportion de chaque ensemble
train_prop = len(X_train) / len(X)
test_prop = len(X_test) / len(X)

print("Proportion de données dans l'ensemble d'entraînement: {:.2f}".format(train_prop))
print("Proportion de données dans l'ensemble de test: {:.2f}".format(test_prop))
```

```
Proportion de données dans l'ensemble d'entraînement: 0.85
Proportion de données dans l'ensemble de test: 0.43
```

1- Regression Logistique

In [26]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE


# Créer le modèle de régression logistique
classifierlogreg = LogisticRegression(penalty='l1', C=0.1, solver='liblinear',random_state = 0)
```

```
# Appliquer la méthode RFE pour sélectionner les variables explicatives les plus importantes
rfe = RFE(classifierlogreg, n_features_to_select=15)
rfe.fit(X_train, Y_train)

# Afficher les variables explicatives sélectionnées
print("Variables explicatives sélectionnées : ")
for i in range(len(X.columns)):
    if rfe.support_[i]:
        print(X.columns[i])

# Afficher les variables du feature selection
print("Variables du feature selection : ")
print(X.columns[rfe.support_])
```

```
Variables explicatives sélectionnées :
item5
item7
item8
item9
item10
item11
item12
item21
item22
item23
item24
item25
item26
item27
item28
Variables du feature selection :
Index(['item5', 'item7', 'item8', 'item9', 'item10', 'item11', 'item12',
       'item21', 'item22', 'item23', 'item24', 'item25', 'item26', 'item27',
       'item28'],
      dtype='object')
```

In [27]:
```
Y_pred = rfe.predict(X_test)
Y_pred
```

Out[27]:
```
array([0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 1])
```

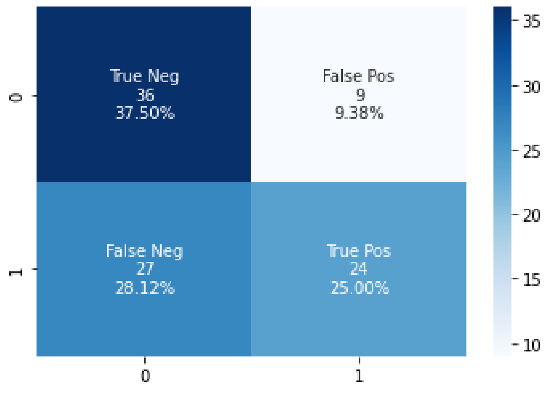
Matrice de confusion

In [28]:
```
from sklearn.metrics import confusion_matrix
CM = confusion_matrix(Y_test, Y_pred)
CM
```

Out[28]:
```
array([[36,  9],
       [27, 24]], dtype=int64)
```

In [29]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                CM.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     CM.flatten()/np.sum(CM)]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(CM, annot=labels, fmt='', cmap='Blues')
```

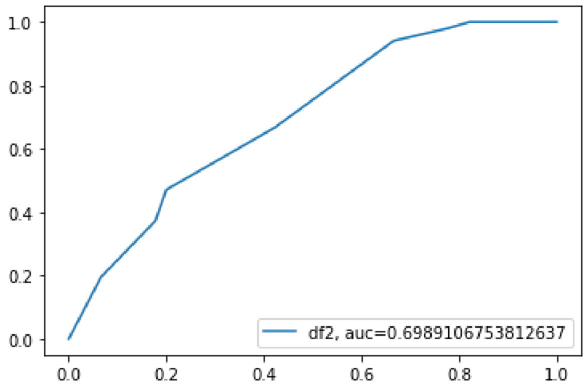
Out[29]:<AxesSubplot:>



Courbe ROC

In [30]:
```
from sklearn import metrics
```

```
y_pred_proba = rfe.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_test,  y_pred_proba)
auc = metrics.roc_auc_score(Y_test, y_pred_proba)
plt.plot(fpr,tpr,label="df2, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



La courbe ROC (Receiver Operating Characteristic) est un graphique du taux de vrais positifs par rapport au taux de faux positifs. Il montre le compromis entre sensibilité et spécificité.

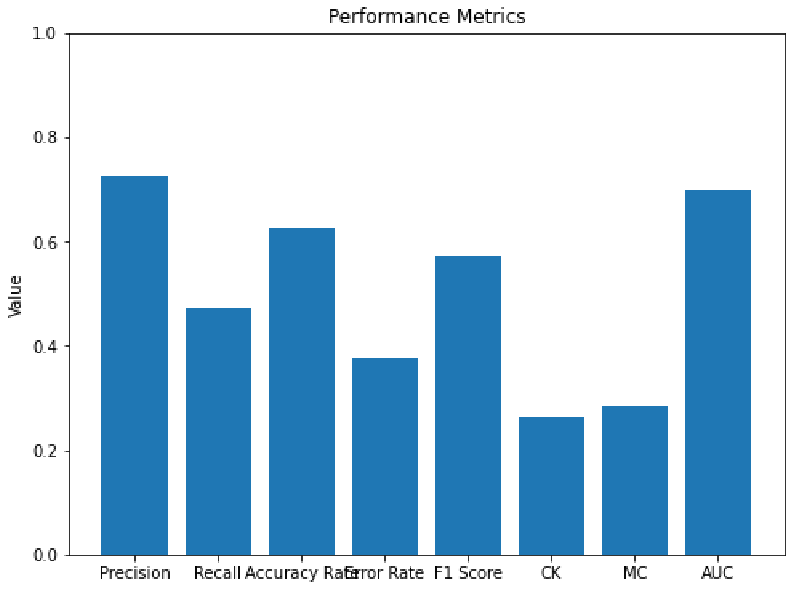
Metrics

In [31]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import matthews_corrcoef
from sklearn import metrics
```

In [32]:
```
Accuracy_Rate = accuracy_score(Y_test, Y_pred)
Error_rate = 1 - Accuracy_Rate
F1_score_logreg = f1_score(Y_test, Y_pred)
precision = precision_score(Y_test, Y_pred)
recall = recall_score(Y_test, Y_pred)
CK = cohen_kappa_score (Y_test,Y_pred)
MC = matthews_corrcoef(Y_test,Y_pred)
auc = metrics.roc_auc_score(Y_test, y_pred_proba)

print("precision : {:.2f}".format(precision))
print("recall : {:.2f}".format(recall))
print("Accuracy rate: ", Accuracy_Rate)
print("Error rate: ", Error_rate)
print("F1_score: ", F1_score_logreg)
print("CK:", CK)
print("MC:", MC)
print("AUC:", auc)
```

```
precision : 0.73
recall : 0.47
Accuracy rate:  0.625
Error rate:  0.375
F1_score:  0.5714285714285714
CK: 0.26436781609195403
MC: 0.28429748028367313
AUC: 0.6989106753812637
```

In [33]:
```
# create a list of metric names and values
metric_names = ["Precision", "Recall", "Accuracy Rate", "Error Rate", "F1 Score", "CK", "MC","AUC"]
metric_values = [precision, recall, Accuracy_Rate, Error_rate, F1_score_logreg, CK, MC,auc]

# create a bar chart
fig, ax = plt.subplots(figsize=(8,6))
ax.bar(metric_names, metric_values)
ax.set_ylabel('Value')
ax.set_ylim([0,1])
ax.set_title('Performance Metrics')
plt.show()
```



2- Decision Tree

In [34]:
```
from sklearn.tree import DecisionTreeClassifier
from boruta import BorutaPy

# Initialisation et entraînement du modèle d'arbre de décision
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, Y_train)
```

Out[34]:
```
DecisionTreeClassifier(random_state=0)
```

In [35]:
```
Y_pred2 = tree.predict(X_test)
Y_pred2
```

Out[35]:
```
array([0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 1])
```

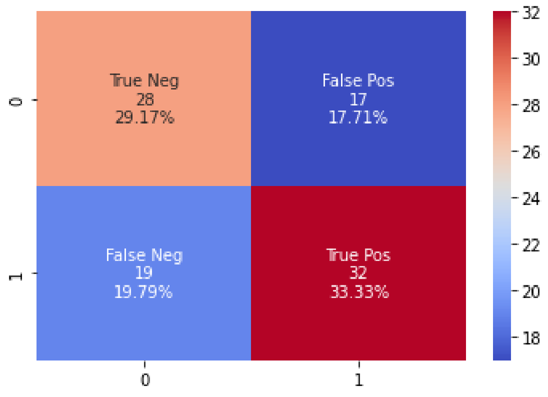
Matrice de confusion

In [36]:
```
CM2 = confusion_matrix(Y_test, Y_pred2)
CM2
```

Out[36]:
```
array([[28, 17],
       [19, 32]], dtype=int64)
```

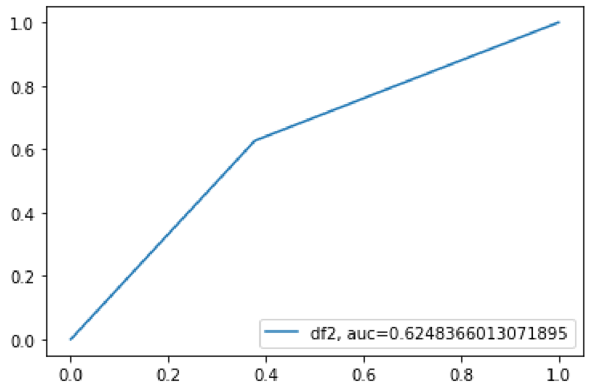
In [37]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                CM2.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     CM2.flatten()/np.sum(CM2)]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(CM2, annot=labels, fmt='', cmap='coolwarm')
```

Out[37]:
```
<AxesSubplot:>
```



Courbe roc

In [38]:
```
y_pred_proba2 = tree.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_test,  y_pred_proba2)
auc = metrics.roc_auc_score(Y_test, y_pred_proba2)
plt.plot(fpr,tpr,label="df2, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

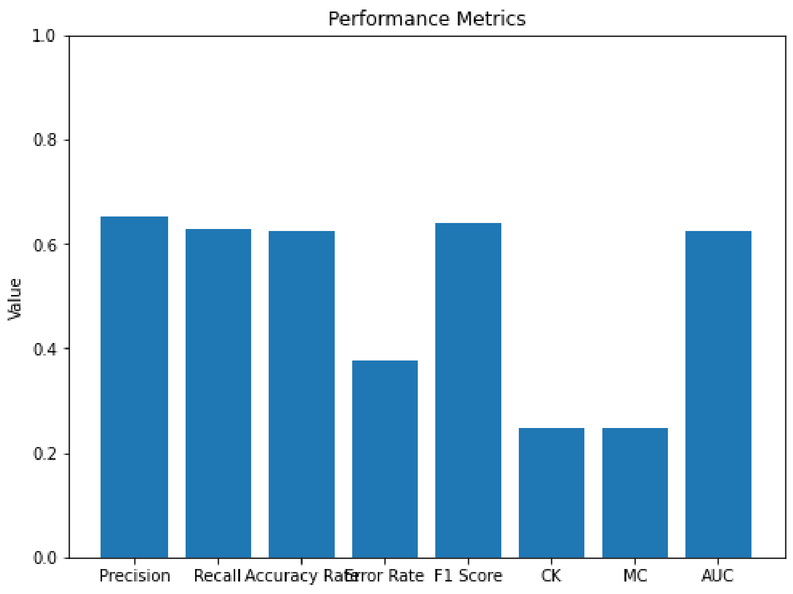


Metrics

In [39]:
```
Accuracy_Rate2 = accuracy_score(Y_test, Y_pred2)
Error_rate2 = 1 - Accuracy_Rate2  # Change variable name to Error_rate2
F1_score_logreg2 = f1_score(Y_test, Y_pred2)
precision2 = precision_score(Y_test, Y_pred2)
recall2 = recall_score(Y_test, Y_pred2)
CK2 = cohen_kappa_score(Y_test, Y_pred2)
MC2 = matthews_corrcoef(Y_test, Y_pred2)
auc2 = metrics.roc_auc_score(Y_test, y_pred_proba2)

print("precision : {:.2f}".format(precision2))
print("recall : {:.2f}".format(recall2))
print("Accuracy rate: ", Accuracy_Rate2)
print("Error rate: ", Error_rate2)
print("F1_score: ", F1_score_logreg2)
print("CK:", CK2)
print("MC:", MC2)
print("AUC:", auc2)
```

```
precision : 0.65
recall : 0.63
Accuracy rate:  0.625
Error rate:  0.375
F1_score:  0.64
CK: 0.24902216427640156
MC: 0.24923917672754117
AUC: 0.6248366013071895
```

In [40]:
```
# create a list of metric names and values
metric_names = ["Precision", "Recall", "Accuracy Rate", "Error Rate", "F1 Score", "CK", "MC","AUC"]
metric_values = [precision2, recall2, Accuracy_Rate2, Error_rate2, F1_score_logreg2, CK2, MC2,auc2]

# create a bar chart
fig, ax = plt.subplots(figsize=(8,6))
ax.bar(metric_names, metric_values)
ax.set_ylabel('Value')
ax.set_ylim([0,1])
ax.set_title('Performance Metrics')
plt.show()
```



3- RandomForest

In [41]:
```
from sklearn.ensemble import RandomForestClassifier
from boruta import BorutaPy

X_train_np = X_train.to_numpy()
Y_train_np= Y_train.to_numpy()
# Création de l'estimateur RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100000, n_jobs=1, max_depth=5)

# Initialisation de BorutaPy
feat_selector = BorutaPy(
    verbose=5,
    estimator=rf,
    n_estimators='auto',
    random_state=1
)

# Exécution de l'algorithme BorutaPy
feat_selector.fit(X_train_np, Y_train_np)

# Get selected features
X_train_selected = feat_selector.transform(X_train.values)
X_test_selected = feat_selector.transform(X_test.values)

# Train a machine learning model using the selected features
model = RandomForestClassifier()
model.fit(X_train_selected, Y_train)

# Get selected features
X_train_selected = feat_selector.transform(X_train.values)

# Affichage des variables sélectionnées
selected_features = pd.DataFrame({'Feature':list(X_train.columns), 'Selected':feat_selector.support_})
selected_features = selected_features[selected_features['Selected']==True]
print(selected_features)
```

```
Iteration:  1 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  2 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  3 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  4 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  5 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  6 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  7 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  8 / 100
Confirmed:  3
Tentative:  7
Rejected:  18
Iteration:  9 / 100
Confirmed:  3
Tentative:  7
Rejected:  18
Iteration:  10 / 100
Confirmed:  3
Tentative:  7
Rejected:  18
Iteration:  11 / 100
Confirmed:  3
Tentative:  7
Rejected:  18
```

```
Iteration:  12 / 100
Confirmed:  3
Tentative:  6
Rejected:  19
Iteration:  13 / 100
Confirmed:  3
Tentative:  6
Rejected:  19
Iteration:  14 / 100
Confirmed:  3
Tentative:  6
Rejected:  19
Iteration:  15 / 100
Confirmed:  3
Tentative:  6
Rejected:  19
Iteration:  16 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  17 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  18 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  19 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  20 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  21 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  22 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  23 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  24 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  25 / 100
Confirmed:  4
Tentative:  4
Rejected:  20
Iteration:  26 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  27 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  28 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  29 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  30 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  31 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  32 / 100
Confirmed:  4
```

```
Tentative:  3
Rejected:  21
Iteration:  33 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  34 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  35 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  36 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  37 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  38 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  39 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  40 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  41 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  42 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  43 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  44 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  45 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  46 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  47 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  48 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  49 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  50 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  51 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  52 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  53 / 100
```

```
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  54 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  55 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  56 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  57 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  58 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  59 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  60 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  61 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  62 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  63 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  64 / 100
Confirmed:  4
Tentative:  3
Rejected:  21
Iteration:  65 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  66 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  67 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  68 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  69 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  70 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  71 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  72 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  73 / 100
Confirmed:  5
Tentative:  2
```

```
Rejected:  21
Iteration:  74 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  75 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  76 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  77 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  78 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  79 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  80 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  81 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  82 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  83 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  84 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  85 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  86 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  87 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  88 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  89 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  90 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  91 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  92 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  93 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  94 / 100
Confirmed:  5
```

```
Tentative:  2
Rejected:  21
Iteration:  95 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  96 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  97 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  98 / 100
Confirmed:  5
Tentative:  2
Rejected:  21
Iteration:  99 / 100
Confirmed:  5
Tentative:  2
Rejected:  21


BorutaPy finished running.

Iteration:  100 / 100
Confirmed:  5
Tentative:  1
Rejected:  21
   Feature  Selected
4    item5      True
16  item17      True
22  item23      True
23  item24      True
24  item25      True
```

MAX_DEPTH

Le paramètre max_depth est un hyperparamètre utilisé dans les arbres de décision et les forêts aléatoires. Il contrôle la profondeur maximale de l'arbre. Plus précisément, max_depth définit le nombre maximum de niveaux que l'arbre de décision peut avoir, c'est-à-dire le nombre maximal de divisions de l'arbre.

Une valeur plus élevée de max_depth permet à l'arbre d'être plus complexe et d'apprendre des modèles plus détaillés à partir des données d'entraînement. Cependant, cela peut également conduire à un surajustement (overfitting) si la profondeur devient trop grande.

Il est courant de régler max_depth en fonction de la taille et de la complexité des données, ainsi que de la quantité d'informations que vous souhaitez extraire de l'arbre. Une valeur par défaut est souvent utilisée, mais il est recommandé d'expérimenter avec différentes valeurs pour trouver celle qui fonctionne le mieux pour votre problème spécifique.

In [42]:
```
Y_pred3 = model.predict(X_test_selected)
print(Y_test)
print(Y_pred3)
```

```
55     0
182    1
92     1
208    0
278    1
      ..
71     0
179    1
231    1
194    0
276    1
Name: Douleur_Centrale, Length: 96, dtype: int32
[0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 0 1 0
 0 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0
 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0]
```

Matrice de confusion

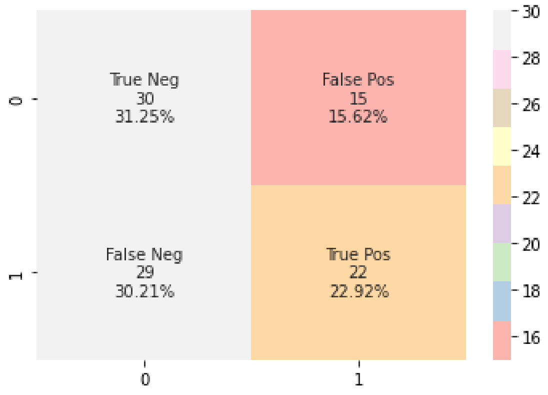
In [43]:
```
CM3 = confusion_matrix(Y_test, Y_pred3)
CM3
```

```
Out[43]:array([[30, 15],
       [29, 22]], dtype=int64)
```

In [44]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ['{0:0.0f}'.format(value) for value in
                CM3.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in
                     CM3.flatten()/np.sum(CM3)]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
```

```
    zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(CM3, annot=labels, fmt='', cmap='Pastel1')
```

```
Out[44]:<AxesSubplot:>
```

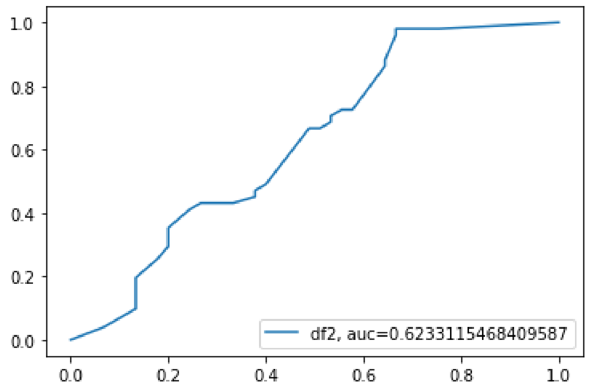


In [45]:
```
print(Y_test.shape)
print(CM3)
```

```
(96,)
[[30 15]
 [29 22]]
```

Courbe roc

In [46]:
```
Y_pred3_proba = model.predict_proba(X_test_selected)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y_test,Y_pred3_proba)
auc = metrics.roc_auc_score(Y_test, Y_pred3_proba)
plt.plot(fpr,tpr,label="df2, auc="+str(auc))
plt.legend(loc=4)  #Y6_pred = model.predict(X_test_selected)
plt.show()
# La courbe roc montre la spécifité et la sensibilité
```



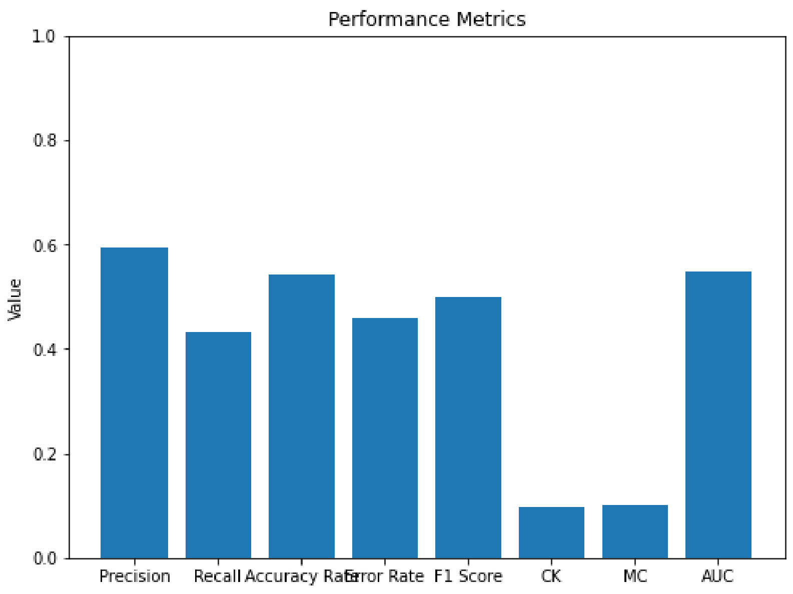
Metrics

In [47]:
```
Accuracy_Rate3 = accuracy_score(Y_test, Y_pred3)
Error_rate3 = 1 - Accuracy_Rate3  # Change variable name to Error_rate2
F1_score_logreg3 = f1_score(Y_test, Y_pred3)
precision3 = precision_score(Y_test, Y_pred3)
recall3 = recall_score(Y_test, Y_pred3)
CK3 = cohen_kappa_score(Y_test, Y_pred3)
MC3 = matthews_corrcoef(Y_test, Y_pred3)
auc3 = metrics.roc_auc_score(Y_test, Y_pred3)


print("precision : {:.2f}".format(precision3))
print("recall : {:.2f}".format(recall3))
print("Accuracy rate: ", Accuracy_Rate3)
print("Error rate: ", Error_rate3)
print("F1_score: ", F1_score_logreg3)
print("CK:", CK3)
print("MC:", MC3)
print("AUC:", auc3)
```

```
precision : 0.59
recall : 0.43
Accuracy rate:  0.5416666666666666
Error rate:  0.45833333333333337
F1_score:  0.5000000000000001
CK: 0.09627727856225932
MC: 0.1005227370952253
AUC: 0.5490196078431373
```

In [48]:
```
# create a list of metric names and values
```

```
metric_names = ["Precision", "Recall", "Accuracy Rate", "Error Rate", "F1 Score", "CK", "MC","AUC"]
metric_values = [precision3, recall3, Accuracy_Rate3, Error_rate3, F1_score_logreg3, CK3, MC3,auc3]

# create a bar chart
fig, ax = plt.subplots(figsize=(8,6))
ax.bar(metric_names, metric_values)
ax.set_ylabel('Value')
ax.set_ylim([0,1])
ax.set_title('Performance Metrics')
plt.show()
```



4- Gradient Boosting

In [49]:
```
from sklearn.ensemble import GradientBoostingClassifier
from boruta import BorutaPy

# Conversion des données en numpy array
X_train_np = X_train.to_numpy()
Y_train_np = Y_train.to_numpy()

# Création de l'estimateur GradientBoostingClassifier
gb = GradientBoostingClassifier(n_estimators=100, max_depth=5)

# Initialisation de BorutaPy
feat_selector = BorutaPy(
    estimator=gb,
    n_estimators='auto',
    verbose=2,
    random_state=1
)

# Exécution de l'algorithme BorutaPy
feat_selector.fit(X_train_np, Y_train_np)

# Affichage des variables sélectionnées
selected_features = pd.DataFrame({'Feature': X_train.columns, 'Selected': feat_selector.support_})
selected_features = selected_features[selected_features['Selected']]

print("Selected features:", selected_features)

# Sélection des colonnes correspondantes aux features sélectionnées
X_train_selected = X_train[selected_features['Feature'].values]
X_test_selected = X_test[selected_features['Feature'].values]

# Entrainement d'un modèle GradientBoosting sur les features sélectionnées
model = GradientBoostingClassifier(n_estimators=100, max_depth=5)
model.fit(X_train_selected, Y_train)

# Evaluation du modèle sur les données de test
accuracy = model.score(X_test_selected, Y_test)
print("Accuracy:", accuracy)
```

```
Iteration:  1 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  2 / 100
Confirmed:  0
Tentative:  28
```

```
Rejected:  0
Iteration:  3 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  4 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  5 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  6 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  7 / 100
Confirmed:  0
Tentative:  28
Rejected:  0
Iteration:  8 / 100
Confirmed:  0
Tentative:  10
Rejected:  18
Iteration:  9 / 100
Confirmed:  1
Tentative:  9
Rejected:  18
Iteration:  10 / 100
Confirmed:  1
Tentative:  9
Rejected:  18
Iteration:  11 / 100
Confirmed:  1
Tentative:  9
Rejected:  18
Iteration:  12 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  13 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  14 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  15 / 100
Confirmed:  3
Tentative:  5
Rejected:  20
Iteration:  16 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  17 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  18 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  19 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  20 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  21 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  22 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  23 / 100
```

```
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  24 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  25 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  26 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  27 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  28 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  29 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  30 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  31 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  32 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  33 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  34 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  35 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  36 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  37 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  38 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  39 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  40 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  41 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  42 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  43 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
```

```
Iteration:  44 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  45 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  46 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  47 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  48 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  49 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  50 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  51 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  52 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  53 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  54 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  55 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  56 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  57 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  58 / 100
Confirmed:  3
Tentative:  3
Rejected:  22
Iteration:  59 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  60 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  61 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  62 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  63 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  64 / 100
Confirmed:  4
Tentative:  2
```

```
Tentative:  2
Rejected:  22
Iteration:  65 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  66 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  67 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  68 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  69 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  70 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  71 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  72 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  73 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  74 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  75 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  76 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  77 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  78 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  79 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  80 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  81 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  82 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  83 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  84 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  85 / 100
```

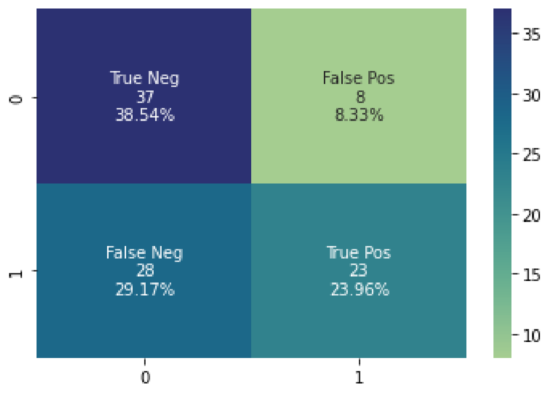
```
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  86 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  87 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  88 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  89 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  90 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  91 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  92 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  93 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  94 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  95 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  96 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  97 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  98 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Iteration:  99 / 100
Confirmed:  4
Tentative:  2
Rejected:  22


BorutaPy finished running.

Iteration:  100 / 100
Confirmed:  4
Tentative:  2
Rejected:  22
Selected features:    Feature  Selected
4    item5      True
18  item19      True
23  item24      True
24  item25      True
Accuracy: 0.625
```

```
In [50]: Y_pred4 = model.predict(X_test_selected)
         print(Y_test)
         print(Y_pred4)
```

```
55     0
182    1
92     1
208    0
278    1
      ..
71     0
179    1
231    1
194    0
276    1
Name: Douleur_Centrale, Length: 96, dtype: int32
[0 0 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0
 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1
 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1]
```

Matrice de confusion

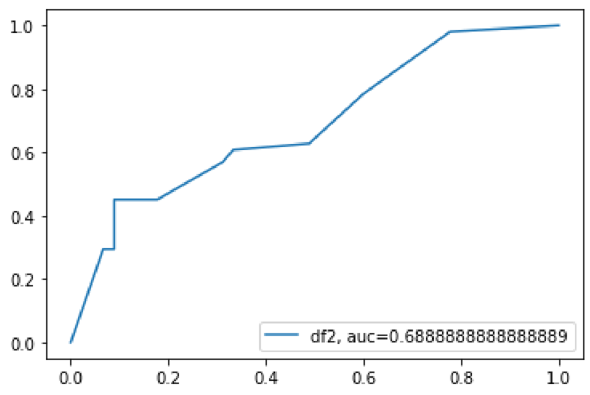
```
In [51]: CM4 = confusion_matrix(Y_test, Y_pred4)
         CM4

Out[51]:array([[37,  8],
               [28, 23]], dtype=int64)

In [52]: group_names = ['True Neg','False Pos','False Neg','True Pos']
         group_counts = ['{0:0.0f}'.format(value) for value in
                         CM4.flatten()]
         group_percentages = ['{0:.2%}'.format(value) for value in
                              CM4.flatten()/np.sum(CM4)]
         labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
                   zip(group_names,group_counts,group_percentages)]
         labels = np.asarray(labels).reshape(2,2)
         sns.heatmap(CM4, annot=labels, fmt='', cmap='crest')

Out[52]:<AxesSubplot:>
```



```
In [53]: print(Y_test.shape)
         print(CM4)

(96,)
[[37  8]
 [28 23]]
```

Courbe roc

```
In [54]: Y_pred_proba4 = model.predict_proba(X_test_selected)[::,1]
         fpr, tpr, _ = metrics.roc_curve(Y_test,Y_pred_proba4)
         auc = metrics.roc_auc_score(Y_test, Y_pred_proba4)
         plt.plot(fpr,tpr,label="df2, auc="+str(auc))
         plt.legend(loc=4)  #Y6_pred = model.predict(X_test_selected)
         plt.show()
         # La courbe roc montre la spécifité et la sensibilité
```



```
In [55]: Accuracy_Rate4 = accuracy_score(Y_test, Y_pred4)
         Error_rate4 = 1 - Accuracy_Rate4  # Change variable name to Error_rate2
```

```
F1_score_logreg4 = f1_score(Y_test, Y_pred4)
precision4 = precision_score(Y_test, Y_pred4)
recall4 = recall_score(Y_test, Y_pred4)
CK4 = cohen_kappa_score(Y_test, Y_pred4)
MC4 = matthews_corrcoef(Y_test, Y_pred4)
auc4 = metrics.roc_auc_score(Y_test, Y_pred4)

print("precision : {:.2f}".format(precision4))
print("recall : {:.2f}".format(recall4))
print("Accuracy rate: ", Accuracy_Rate4)
print("Error rate: ", Error_rate4)
print("F1_score: ", F1_score_logreg4)
print("CK:", CK4)
print("MC:", MC4)
print("AUC:", auc4)
```

```
precision : 0.74
recall : 0.45
Accuracy rate:  0.625
Error rate:  0.375
F1_score:  0.5609756097560976
CK: 0.26624203821656056
MC: 0.29156720298061406
AUC: 0.6366013071895424
```

ACP: Analyse en Composante Principale

In [58]:
```
! pip install nbconvert[webpdf]
```

```
Requirement already satisfied: nbconvert[webpdf] in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (6.4.4)
Requirement already satisfied: jupyter-core in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (5.3.0)
Requirement already satisfied: traitlets>=5.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (5.9.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (1.5.0)
Requirement already satisfied: defusedxml in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (0.7.1)
Requirement already satisfied: jinja2>=2.4 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (3.0.3)
Requirement already satisfied: jupyterlab-pygments in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (0.1.2)
Requirement already satisfied: testpath in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (0.5.0)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (0.8.4)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (0.4)
Requirement already satisfied: bleach in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (4.1.0)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (0.5.13)
Requirement already satisfied: pygments>=2.4.1 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (2.11.2)
Requirement already satisfied: beautifulsoup4 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (4.11.1)
Requirement already satisfied: nbformat>=4.4 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbconvert[webpdf]) (5.3.0)
Collecting pyppeteer<1.1,>=1
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jinja2>=2.4->nbconvert[webpdf]) (2.
1.1)
Requirement already satisfied: nest-asyncio in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert[webpd
f]) (1.5.5)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconver
t[webpdf]) (7.3.4)
Requirement already satisfied: tornado>=6.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=
0.5.0->nbconvert[webpdf]) (6.1)
Requirement already satisfied: pyzmq>=23.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=
0.5.0->nbconvert[webpdf]) (23.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jupyter-client>=6.1.5->nbclient<
0.6.0,>=0.5.0->nbconvert[webpdf]) (2.8.2)
Requirement already satisfied: platformdirs>=2.5 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jupyter-core->nbconvert[webpdf]) (3.
5.0)
Requirement already satisfied: pywin32>=300 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jupyter-core->nbconvert[webpdf]) (302)
Requirement already satisfied: jsonschema>=2.6 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbformat>=4.4->nbconvert[webpdf])
(4.4.0)
Requirement already satisfied: fastjsonschema in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from nbformat>=4.4->nbconvert[webpdf]) (2
.15.1)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jsonsche
ma>=2.6->nbformat>=4.4->nbconvert[webpdf]) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from jsonschema>=2.6->nbformat>=4.4->nbc
onvert[webpdf]) (21.4.0)
Requirement already satisfied: certifi>=2021 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[webpdf])
(2022.12.7)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[
webpdf]) (1.26.11)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from pyppeteer<1.1,>=1->nbconve
rt[webpdf]) (4.11.3)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[w
ebpdf]) (4.64.1)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from pyppeteer<1.1,>=1->nbconvert[
webpdf]) (1.4.4)
Collecting websockets<11.0,>=10.0
  Downloading websockets-10.4-cp39-cp39-win_amd64.whl (101 kB)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: zipp>=0.5 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from importlib-metadata>=1.4->pyppeteer<1.1,>=
1->nbconvert[webpdf]) (3.8.0)
Requirement already satisfied: six>=1.5 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from python-dateutil>=2.8.2->jupyter-client>=6.1.5-
>nbclient<0.6.0,>=0.5.0->nbconvert[webpdf]) (1.16.0)
Requirement already satisfied: colorama in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from tqdm<5.0.0,>=4.42.1->pyppeteer<1.1,>=1->
nbconvert[webpdf]) (0.4.5)
Requirement already satisfied: soupsieve>1.2 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from beautifulsoup4->nbconvert[webpdf]) (2.
3.1)
Requirement already satisfied: packaging in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from bleach->nbconvert[webpdf]) (21.3)
Requirement already satisfied: webencodings in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from bleach->nbconvert[webpdf]) (0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\aliah\anaconda3\envs\pythonproject\lib\site-packages (from packaging->bleach->nbconv
ert[webpdf]) (3.0.4)
Installing collected packages: websockets, pyee, pyppeteer
Successfully installed pyee-8.2.2 pyppeteer-1.0.2 websockets-10.4
```