Rapport 2 : Traitement d’images

Thème : AVC

PLAN

1- Traitement d'image sous Python

1.1 Packages utilisés

```
[ ]  import pydicom
     import pydicom as dicom
     import numpy as np
     import matplotlib.pyplot as plt
     import exposure
     import skimage
     from skimage import exposure
     import numpy as np
     import cv2
     import scipy
     from scipy import ndimage
     from skimage.color import rgb2gray
```

1.2 Extraction de différentes données

```
dataset=dicom.read_file("case4a_001.dcm")
print(dataset)
```

```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 198
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: MR Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.3.46.670589.11.0.0.11.4.2.0.5526.5.1968.2005110211361181738
(0002, 0010) Transfer Syntax UID                 UI: Implicit VR Little Endian
(0002, 0012) Implementation Class UID            UI: 1.2.276.0.50.20060201.4.1
(0002, 0013) Implementation Version Name         SH: 'JIVEX_TK_41'
-------------------------------------------------
(0008, 0005) Specific Character Set              CS: 'ISO_IR 100'
(0008, 0008) Image Type                          CS: ['ORIGINAL', 'PRIMARY', 'M_SE', 'M', 'SE']
(0008, 0012) Instance Creation Date              DA: '20051102'
(0008, 0013) Instance Creation Time              TM: '115144'
(0008, 0014) Instance Creator UID                UI: 1.3.46.670589.11.5526.5
(0008, 0016) SOP Class UID                       UI: MR Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.2.276.0.50.192168001099.8687553.14547392.48
(0008, 0020) Study Date                          DA: '20010101'
(0008, 0021) Series Date                         DA: '20070511'
(0008, 0022) Acquisition Date                    DA: '20070511'
(0008, 0023) Content Date                        DA: '20070511'
(0008, 0030) Study Time                          TM: '114244'
(0008, 0031) Series Time                         TM: '114244'
(0008, 0032) Acquisition Time                    TM: '114244'
(0008, 0033) Content Time                        TM: '114244'
(0008, 0050) Accession Number                    SH: '11788765469991'
(0008, 0060) Modality                            CS: 'MR'
(0008, 0070) Manufacturer                        LO: 'Philips Medical Systems'
(0008, 0080) Institution Name                    LO: 'Anonymized Hospital'
```

```
(0008, 0081) Institution Address                ST: 'Anonymized Address'
(0008, 0090) Referring Physician's Name         PN: 'Dr. Anonymous'
(0008, 1010) Station Name                       SH: 'Any Station'
(0008, 1030) Study Description                  LO: 'MRT Schädel'
(0008, 103e) Series Description                 LO: 'Brain      *T2/GRASE'
(0008, 1040) Institutional Department Name      LO: ''
(0008, 1080) Admitting Diagnoses Description    LO: ''
(0008, 1090) Manufacturer's Model Name          LO: 'Intera'
(0008, 1111)  Referenced Performed Procedure Step Sequence  1 item(s) ----
   (0008, 0005) Specific Character Set             CS: 'ISO_IR 100'
   (0008, 0012) Instance Creation Date             DA: '20051102'
   (0008, 0013) Instance Creation Time             TM: '115145'
   (0008, 0014) Instance Creator UID               UI: 1.3.46.670589.11.5526.5
   (0008, 1150) Referenced SOP Class UID           UI: Modality Performed Procedure Step SOP Class
   (0008, 1155) Referenced SOP Instance UID        UI: 1.3.46.670589.11.0.0.11.4.2.0.5526.5.1208.2005110211202042266
   (0020, 0013) Instance Number                    IS: '0'
   ---------
(0008, 1140)  Referenced Image Sequence  3 item(s) ----
   (0008, 1150) Referenced SOP Class UID           UI: MR Image Storage
   (0008, 1155) Referenced SOP Instance UID        UI: 1.3.46.670589.11.0.0.11.4.2.0.5526.5.1968.2005110211302184712
   ---------
   (0008, 1150) Referenced SOP Class UID           UI: MR Image Storage
   (0008, 1155) Referenced SOP Instance UID        UI: 1.3.46.670589.11.0.0.11.4.2.0.5526.5.1968.2005110211302179709
   ---------
   (0008, 1150) Referenced SOP Class UID           UI: MR Image Storage
   (0008, 1155) Referenced SOP Instance UID        UI: 1.3.46.670589.11.0.0.11.4.2.0.5526.5.1968.2005110211302181710
   ---------
(0010, 0010) Patient's Name                     PN: 'Fall 4'
(0010, 0020) Patient ID                         LO: '11788765469993'
(0010, 0030) Patient's Birth Date               DA: '19000101'
(0010, 0032) Patient's Birth Time               TM: '0000'
(0010, 0040) Patient's Sex                      CS: 'O'
(0010, 1030) Patient's Weight                   DS: '75.0'
(0010, 2000) Medical Alerts                     LO: ''
(0010, 2110) Allergies                          LO: ''
(0010, 2160) Ethnic Group                       SH: ''
(0010, 2180) Occupation                         SH: ''
(0010, 21b0) Additional Patient History         LT: ''
(0010, 21c0) Pregnancy Status                   US: 4
(0010, 4000) Patient Comments                   LT: ''
(0018, 0020) Scanning Sequence                  CS: 'SE'
(0018, 0021) Sequence Variant                   CS: 'SK'
(0018, 0022) Scan Options                       CS: 'FS'
(0018, 0023) MR Acquisition Type                CS: ''
(0018, 0024) Sequence Name                      SH: ''
(0018, 0050) Slice Thickness                    DS: '5.0'
(0018, 0080) Repetition Time                    DS: '5286.37451171875'
(0018, 0081) Echo Time                          DS: '100.0'
(0018, 0082) Inversion Time                     DS: '0.0'
(0018, 0083) Number of Averages                 DS: '4.0'
(0018, 0084) Imaging Frequency                  DS: '63.892263'
(0018, 0085) Imaged Nucleus                     SH: '1H'
(0018, 0086) Echo Number(s)                     IS: '1'
(0018, 0087) Magnetic Field Strength            DS: '1.5'
(0018, 0088) Spacing Between Slices             DS: '6.0'
```

```
Number of Phase Encoding Steps      IS: '237'
Echo Train Length                   IS: '9'
Percent Sampling                    DS: '50.0'
Percent Phase Field of View         DS: '79.6875'
Device Serial Number                LO: '05526'
Software Versions                   LO: ['NT 10.3.1', 'PIIM V2.1.4.1 MIMIT MCS']
Protocol Name                       LO: '*T2/GRASE'
Low R-R Value                       IS: '0'
High R-R Value                      IS: '0'
Intervals Acquired                  IS: '0'
Intervals Rejected                  IS: '0'
Heart Rate                          IS: '0'
Reconstruction Diameter             DS: '230.0'
Receive Coil Name                   SH: 'Head'
Transmit Coil Name                  SH: 'B'
Acquisition Matrix                  US: [0, 512, 237, 0]
In-plane Phase Encoding Direction   CS: 'ROW'
Flip Angle                          DS: '90.0'
Patient Position                    CS: 'HFS'
Study Instance UID                  UI: 1.2.276.0.50.192168001099.8687553.14547392.4
Series Instance UID                 UI: 1.2.276.0.50.192168001099.8687553.14547392.34
Study ID                            SH: '11788765470094'
Series Number                       IS: '301'
Acquisition Number                  IS: '3'
Instance Number                     IS: '14'
Image Position (Patient)            DS: [-103.25500191355, -104.36103178963, 91.9502137311336]
(0020, 0032) Image Position (Patient)          DS: [-103.25500191355, -104.36103178963, 91.9502137311336]
(0020, 0037) Image Orientation (Patient)       DS: [0.99183012796786, -1.0599023325E-9, -0.1275656586046, -0.0341314271334, 0.96354128711252, -0.2653737622889]
(0020, 0052) Frame of Reference UID            UI: 1.3.46.670589.11.0.0.11.4.2.0.5526.5.5036.2005110211295229361
Laterality                          CS: ''
Temporal Position Identifier        IS: '1'
Number of Temporal Positions        IS: '1'
Position Reference Indicator        LO: ''
Slice Location                      DS: '78.0000002901833'
Image Comments                      LT: ''
Samples per Pixel                   US: 1
Photometric Interpretation          CS: 'MONOCHROME2'
Rows                                US: 512
Columns                             US: 512
Pixel Spacing                       DS: [0.44921875, 0.44921875]
Pixel Aspect Ratio                  IS: [1, 1]
Bits Allocated                      US: 16
Bits Stored                         US: 12
High Bit                            US: 11
Pixel Representation                US: 0
Smallest Image Pixel Value          US: 0
Largest Image Pixel Value           US: 730
Window Center                       DS: '346.919723638756'
Window Width                        DS: '692.492186214838'
Lossy Image Compression             CS: '00'
Requesting Physician                PN: 'RequestingPhysician'
Requesting Service                  LO: 'RequestingService'
Requested Procedure Description     LO: 'MRT Schädel'
Requested Contrast Agent            LO: ''
Study Comments                      LT: 'MRT Schädel'
Special Needs                       LO: ''
Patient State                       LO: ''
```

```
 Performed Station AE Title          AE: 'MR_STORE_EXP'
 Performed Station Name              SH: ''
 Performed Location                  SH: ''
 Performed Procedure Step Start Date DA: '20051102'
 Performed Procedure Step Start Time TM: '112742'
 Performed Procedure Step End Date   DA: '20051102'
 Performed Procedure Step End Time   TM: '112742'
 Performed Procedure Step Status     CS: ''
 Performed Procedure Step ID         SH: '184159220'
 Performed Procedure Step Descriptio LO: 'MRT Schädel'
 Performed Procedure Type Descriptio LO: ''
  Request Attributes Sequence  1 item(s) ----
07) Scheduled Procedure Step Descriptio LO: 'MRT Schädel'
08)  Scheduled Protocol Code Sequence  0 item(s) ----
09) Scheduled Procedure Step ID         SH: '0003717687'
01) Requested Procedure ID              SH: '0003717687'
12) Private Creator                     LO: 'Philips MR Imaging DD 003'
13) [Unknown]                           UL: 0

 Comments on the Performed Procedure ST: 'MRT Schädel'
  Film Consumption Sequence  0 item(s) ----
 Requested Procedure ID              SH: '0003717687'
 Reason for the Requested Procedure  LO: ''
 Requested Procedure Priority        SH: ''
 Patient Transport Arrangements      LO: ''
 Requested Procedure Location        LO: ''
 Names of Intended Recipients of Res PN: ''
 Requested Procedure Comments        LT: ''
 Reason for the Imaging Service Requ LO: ''
Reason for the Imaging Service Requ LO: ''
Issue Date of Imaging Service Reque DA: ''
Issue Time of Imaging Service Reque TM: ''
Placer Order Number / Imaging Servi SH: ''
Filler Order Number / Imaging Servi SH: ''
Order Entered By                    PN: ''
Order Enterer's Location            SH: ''
Order Callback Phone Number         SH: ''
Imaging Service Request Comments    LT: ''
Private tag data                    UN: b'\x00\x00\x00\x00'
Private tag data                    UN: b'14'
Pixel Data                          OW: Array of 524288 elements
```

```
(2001, 1021) [SPIR]                                CS: 'N'
(2001, 1022) [Water Fat Shift]                     FL: 1.4598678350448608
(2001, 1023) [Flip Angle Philips]                  DS: "90.0"
(2001, 1060) [Number of Stacks]                    SL: 1
(2001, 107b) [Acquisition Number]                  IS: "4"
(2001, 1081) [Number of Dynamic Scans]             IS: "1"
(2001, 10f1) [Prospective Motion Correction]       FL: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
(2005, 0010) Private Creator                       LO: 'Philips MR Imaging DD 001'
(2005, 0011) Private Creator                       LO: 'Philips MR Imaging DD 002'
(2005, 0012) Private Creator                       LO: 'Philips MR Imaging DD 003'
(2005, 0013) Private Creator                       LO: 'Philips MR Imaging DD 004'
(2005, 0014) Private Creator                       LO: 'Philips MR Imaging DD 005'
(2005, 1020) [Number of Chemical Shift]            SL: 0
(2005, 10a1) [Syncra Scan Type]                    CS: 'SENSE'
(2005, 10b0) [Diffusion Direction RL]              FL: 0.0
(2005, 10b1) [Diffusion Direction AP]              FL: 0.0
(2005, 10b2) [Diffusion Direction FH]              FL: 0.0
(2050, 0020) Presentation LUT Shape                CS: 'IDENTITY'
(7fe0, 0010) Pixel Data                            OW: Array of 131072 elements
```

Caractéristiques de l'image :

- Imagerie Par Résonance Magnétique IRM
- Type D'examen : MR : Magnetic Resonance
- Dispositif D'acquisition : Aimant Qui Permet De Créer Des Champs Magnétiques
- Partie Du Corps Examiné : CERVEAU
- Sexe du Patient : Masculin
- Le poids du patient : 75Kg
- Identifiant du patient : 'AVC Dx-01-0001'
- Séquence de numérisation : 'SE' : Echo de rotation
- Désignateur de schéma de codage : 'DCM'
- Fabricant : 'Philips Medical Systems'
- Type d'acquisition MR : '2D'
- Description de l'étude : 'MRI PROSTATE WITH AND WITHOUT CONTRAST'
- Contraste : T2, L'eau Apparaît Hyper Intense (Couleur Claire) Et La Graisse Un Peu Plus Sombre Que L'eau
- Formation Des Images : Image En Coupe
- Position de l'image : [-102.04013347625, -0.8545150756835, 41.5915451049804]
- Temps D'écho : 120s
- Temps De Répétition : "3000.0"
- Nombre D'écho : 1
- Nombre d'étapes de codage de phase : '189'
- Les Lignes : 512
- Les Colonnes : 512
- Espacement des pixels : [0.703125, 0.703125]
- Épaisseur de tranche : " 3.0 "
- Dimension en pixel : 512*512
- Interprétation photométrique : 'MONOCHROME2' où les données de pixels sont représentées comme un seul plan d'image monochrome, et la valeur d'échantillonnage minimale est destinée à être affichée sous forme d'informations noires.
- Echantillons par pixel : 1 car cette image est monochrome (ça peut être 3 si par exemple l'espace colorimétrique était RVB).

- **1.3 Extraction de différentes données**

```
print(dataset.Rows)
print(dataset.Columns)
```

512

512

La méthode lit le fichier DICOM et retourne numpy.array () des valeurs de pixels .Elle sert à transformer l'image en matrice de pixel. L'entrée de cette méthode est un chemin vers un fichier appelé "filename".

```
def dicom_to_array(filename):
    d = dicom.read_file(filename)
    a = d.pixel_array
    return np.array(a)

a1 = dicom_to_array("case4a_001.dcm")
print(a1.size)
```

```
262144
```

La commande print(a1.shape) : Affiche la largeur et la hauteur en pixel.

```
print(a1.shape)
```

```
(512, 512)
```

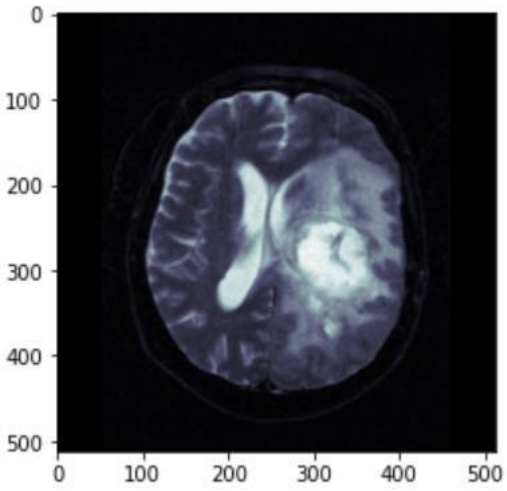
```
print(np.ndarray.min(a1))
print(np.ndarray.max(a1))
```

```
0
730
```

1.4 Visualisation d'image

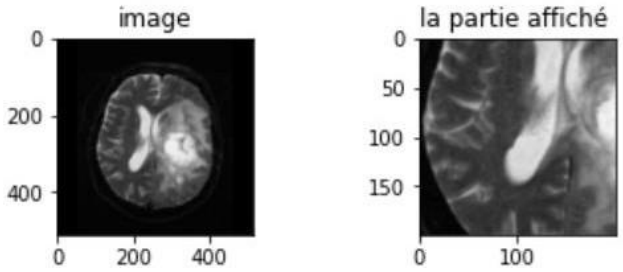
1.4.1 Visualisation d'une seule image

```
[ ]  plt.imshow(dataset.pixel_array,plt.cm.bone)
     plt.show()
```



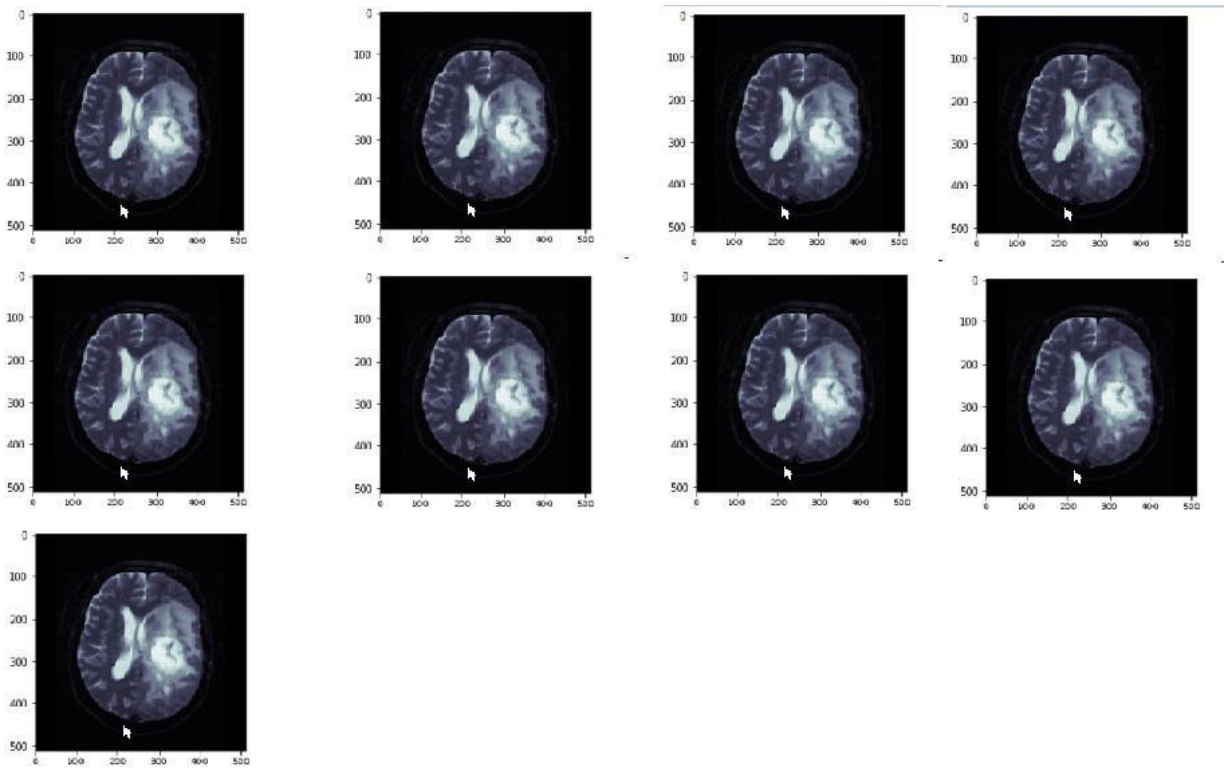
1.4.2 Affichage d'une partie de l'image

```
[ ]  part=a1[200:400,100:300]
     plt.subplot(2,2,1),plt.imshow(a1,cmap = 'gray')
     plt.title('image')
     plt.subplot(2,2,2),plt.imshow(part,cmap = 'gray')
     plt.title('la partie affiché ')

     plt.show()
```



1.4.3 Visualisation de quelques images du fichier DICOM

```
[ ]  folder = 'case4a_001.dcm' #folder where photos are stored
     fig=plt.figure(figsize=(8, 8))
     columns=4
     rows=5
for i in range(1,10):

    # define subplot 1-0.dcm

    #x= folder + '1-0' + str(i) + '.dcm'
    ds = pydicom.filereader.dcmread(folder)
    fig.add_subplot(rows,columns,i)
    # load image pixels
    plt.imshow(ds.pixel_array, cmap=plt.cm.bone)
plt.show()
plt.show()
```

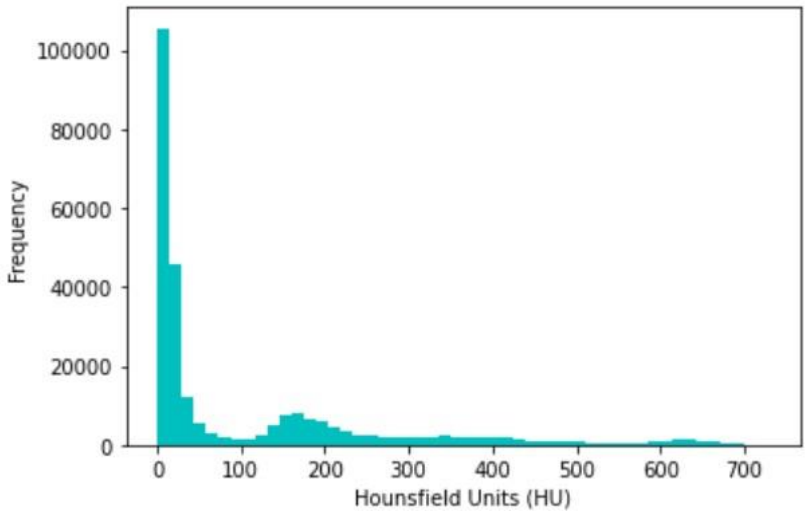


1.5 Histogramme

L'histogramme est une autre façon de comprendre l'image. En regardant l'histogramme d'une image, on obtient une intuition sur le contraste, la luminosité, la distribution d'intensité, etc. de cette image. L'égalisation de l'histogramme améliore le contraste et donc les détails de l'image deviennent plus clairs que l'image originale.

1.5.1 Histogramme Hounsfield

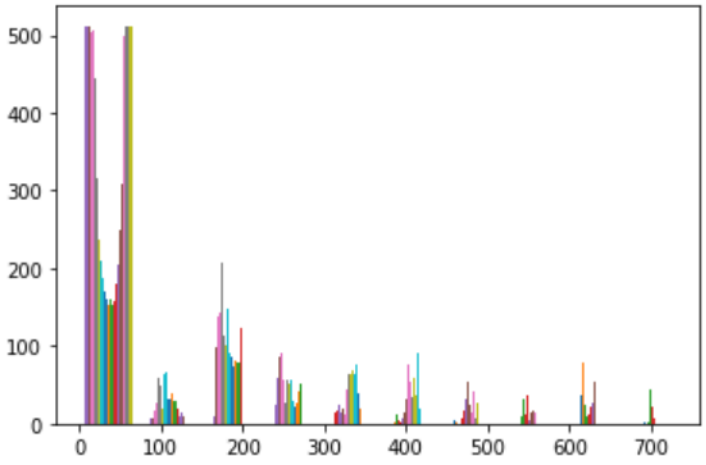
```
plt.hist(a1.flatten(), bins=50, color='c')
plt.xlabel("Hounsfield Units (HU)")
plt.ylabel("Frequency")
plt.show()
```



1.5.2 Histogramme pixel

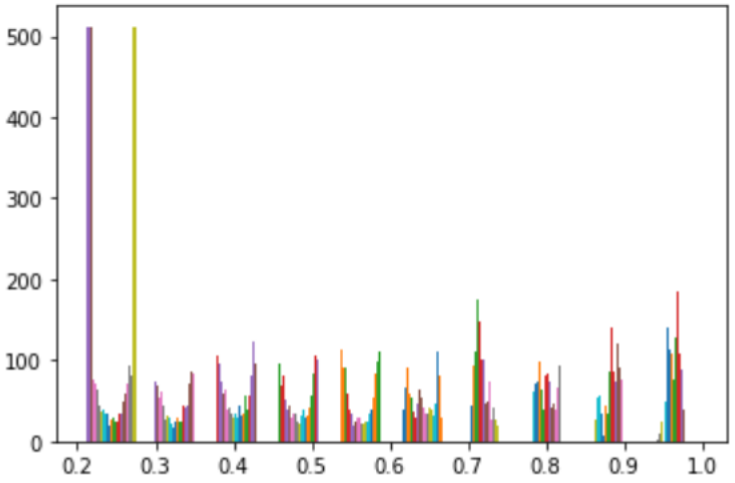
```
[ ]  hist,bins = np.histogram(a1, bins=256)
     plt.figure()
     plt.hist(a1)
```

```
(array([[512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        ...,
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.]]),
 array([  0.,  73., 146., 219., 292., 365., 438., 511., 584., 657., 730.]),
 <a list of 512 Lists of Patches objects>)
```



```
[ ]  a1_eq = skimage.exposure.equalize_hist(a1)
     hist_eq,bins_eq = np.histogram(a1_eq, bins=256)
     plt.figure() # créer une nouvelle figure
     plt.hist(a1_eq)
```

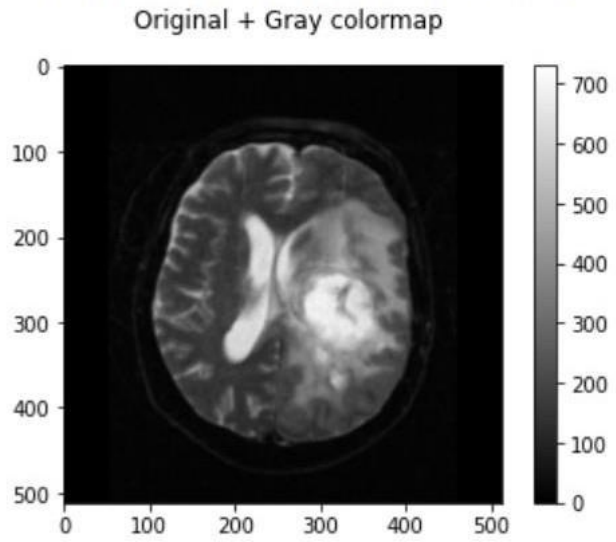
```
(array([[512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        ...,
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.],
        [512.,   0.,   0., ...,   0.,   0.,   0.]]),
 array([0.20436859, 0.28393173, 0.36349487, 0.44305801, 0.52262115,
        0.6021843 , 0.68174744, 0.76131058, 0.84087372, 0.92043686,
        1.        ]),
 <a list of 512 Lists of Patches objects>)
```



Nous allons tracer l'image originale et son égalisation d'histogramme avec la palette de couleurs grise.

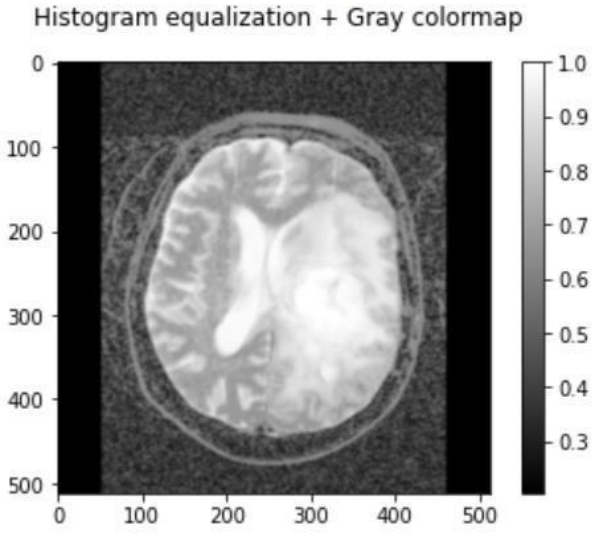
```
[ ]  fig1  = plt.figure()
     plt.imshow(a1, cmap="gray", interpolation="bicubic")
     plt.colorbar()
     fig1.suptitle("Original + Gray colormap", fontsize=12)
```

```
Text(0.5, 0.98, 'Original + Gray colormap')
```




```
[ ]  fig2 = plt.figure()
     plt.imshow(a1_eq, cmap="gray", interpolation="bicubic")
     plt.colorbar()
     fig2.suptitle("Histogram equalization + Gray colormap", fontsize=12)
```

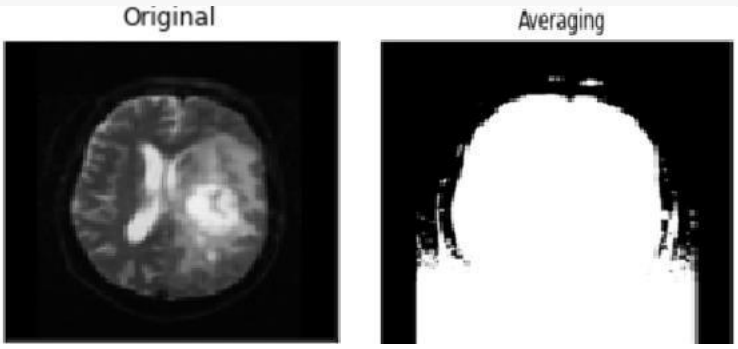
```
Text(0.5, 0.98, 'Histogram equalization + Gray colormap')
```



1.6 Filtrage de l'image

1.6.1 Convolution 2D

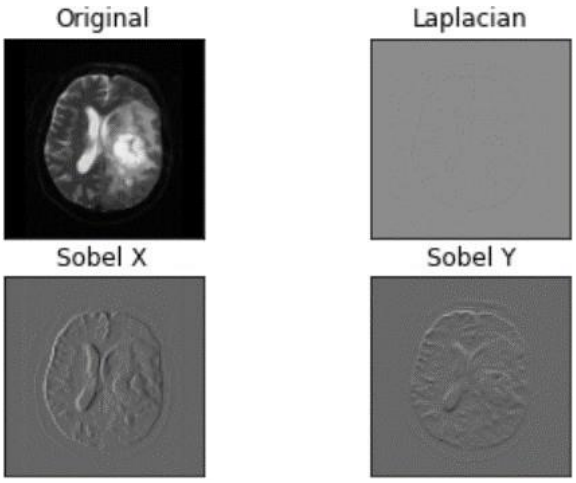
```
#Convolution 2D
kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(a1,-1,kernel)
plt.subplot(121),plt.imshow(a1,'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(dst,'gray'),plt.title('Averaging')
plt.xticks([]), plt.yticks([])
plt.show()
```



1.6.2 Filtre Sobel et Laplacian

Entrée [13]:
```
#Filtre sobel et laplacian
laplacian = cv2.Laplacian(a1,cv2.CV_64F)
sobelx = cv2.Sobel(a1,cv2.CV_64F,1,0,ksize=5)
sobely = cv2.Sobel(a1,cv2.CV_64F,0,1,ksize=5)

plt.subplot(2,2,1),plt.imshow(a1,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
```

Out[13]:
```
(Text(0.5, 1.0, 'Sobel Y'), ([], []), ([], []))
```

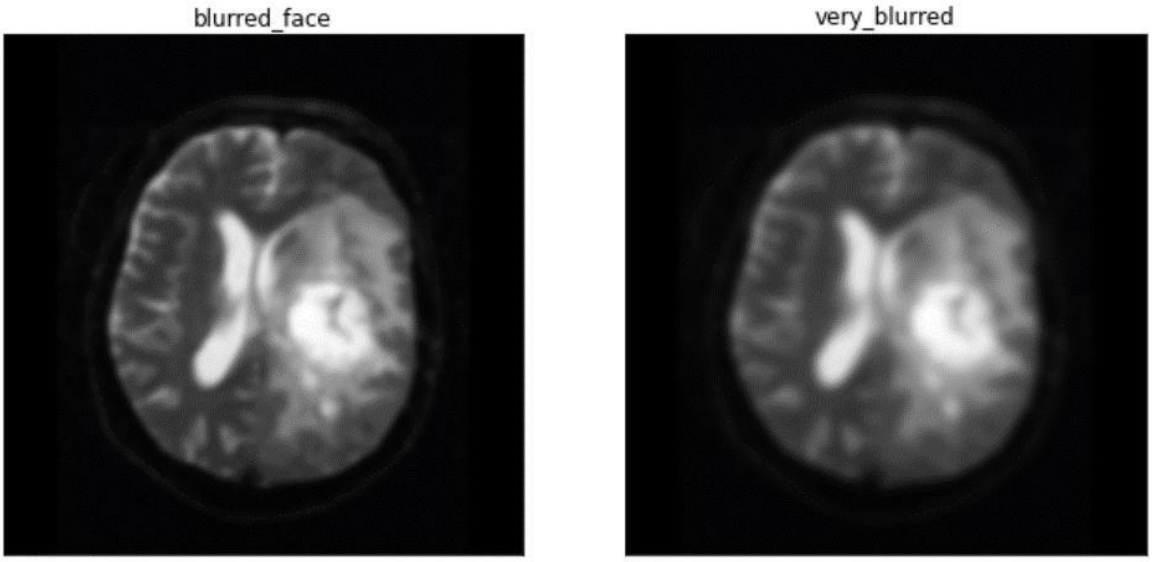


1.6.3 Filtre Gaussien

Dans cette partie, nous avons besoin d'utiliser les packages :

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter
import pylab
import matplotlib.cm as cm
from skimage.color import rgb2gray
from scipy import ndimage
from skimage.io import imread
from skimage.color import rgb2gray
from tkinter import Tk, W, E
from tkinter.ttk import Frame, Button, Entry, Style
```

```
#Filtre gaussien
blurred_face = ndimage.gaussian_filter(a1, sigma=3)
very_blurred = ndimage.gaussian_filter(a1, sigma=5)
plt.figure(figsize=(11, 6))
plt.subplot(121), plt.imshow(blurred_face, cmap='gray')
plt.title('blurred_face'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(very_blurred, cmap='gray')
plt.title('very_blurred'), plt.xticks([]), plt.yticks([])
plt.show()
```



1.7 Morphologie mathématique

1.7.1 Erosion

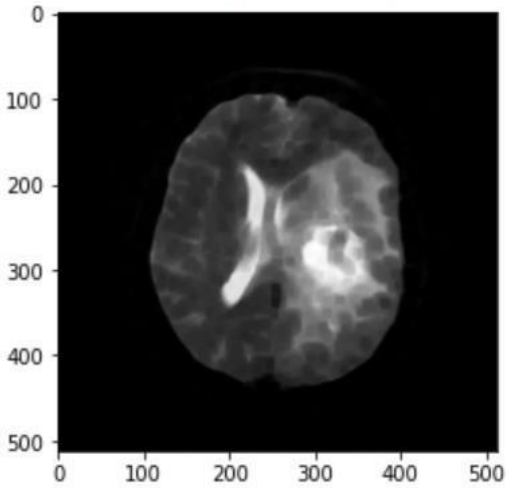
L'érosion érode les limites de l'objet du premier plan. Alors qu'est-ce que ça fait? Le noyau glisse à travers l'image (comme dans la convolution 2D). Un pixel de l'image d'origine (1 ou 0) sera considéré comme 1 uniquement si tous les pixels sous le noyau sont 1, sinon il est érodé (mis à zéro).

Donc, ce qui se passe, c'est que tous les pixels proches de la limite seront rejetés en fonction de la taille du noyau. Ainsi, l'épaisseur ou la taille de l'objet au premier plan diminue ou simplement la zone blanche diminue dans l'image. Il est utile pour supprimer les petits bruits blancs.

L'érosion donc consiste à chercher tous les pixels pour lesquels l'élément structurant centré sur ce pixel touche l'extérieur de la structure. Le résultat est une structure rognée.

```
a1 = dicom_to_array("case4a_001.dcm")

kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(a1,kernel,iterations = 2)
plt.imshow(erosion,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3a91efe6d0>
```



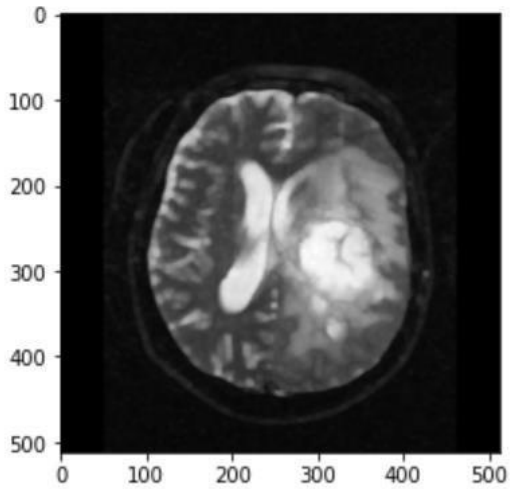
1.7.2 Dilatation

C'est juste l'opposé de l'érosion. Ici, un élément pixel est égal à «1» si au moins un pixel sous le noyau est égal à «1». Ainsi, il augmente la zone blanche de l'image ou la taille de l'objet au premier plan augmente. Normalement, dans des cas comme l'élimination du bruit, l'érosion est suivie d'une dilatation. Parce que l'érosion supprime les bruits blancs, mais elle rétrécit aussi notre objet. Alors on le dilate. Puisque le bruit est parti, notre zone d'objet augmente. Il est également utile pour joindre des parties cassées d'un objet.

La dilatation morphologique donc consiste à déplacer l'élément structurant sur chaque pixel de l'image, et à regarder si l'élément structurant touche la structure d'intérêt.

```
dilation = cv2.dilate(a1,kernel,iterations = 1)
plt.imshow(dilation,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3a91e4dd50>
```

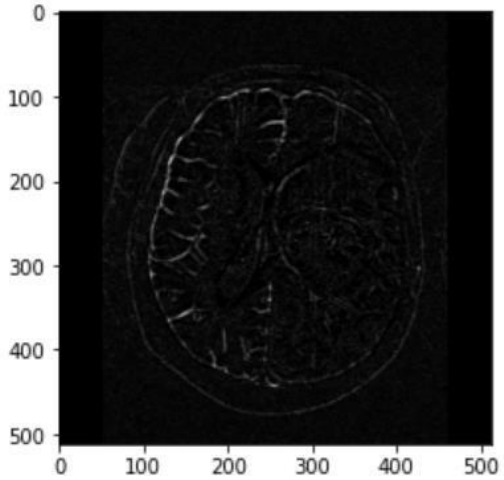


2.7.3 Ouverture

L'ouverture n'est qu'un autre nom d'**érosion suivi de dilatation**. Il est utile pour supprimer le bruit, comme nous l'avons expliqué ci-dessus. Ici, nous utilisons la fonction, **cv2.morphologyEx ()**.

```
opening = cv2.morphologyEx(a1, cv2.MORPH_OPEN, kernel)
plt.imshow(opening,cmap='gray')
```

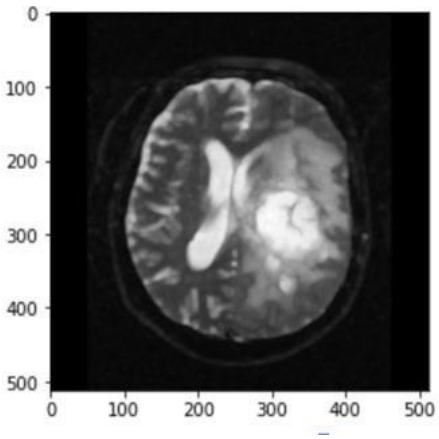
```
<matplotlib.image.AxesImage at 0x7f3a91e35710>
```



1.7.4 Fermeture

La fermeture est l'inverse de l'ouverture, la **dilatation suivie de l'érosion.** Il est utile pour fermer de petits trous à l'intérieur des objets de premier plan ou de petits points noirs sur l'objet.

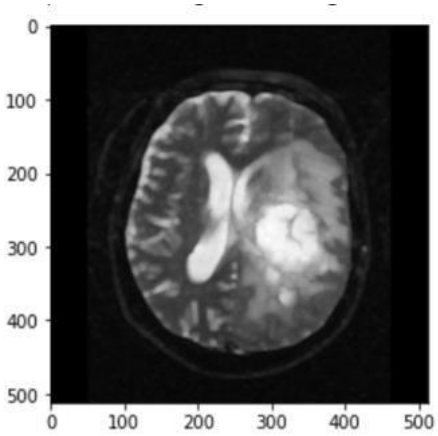
```
#Closing
closing = cv2.morphologyEx(a1, cv2.MORPH_CLOSE, kernel)
plt.imshow(closing,cmap='gray')
```



1.7.5 Gradient morphologique

C'est la différence entre la dilatation et l'érosion d'une image. Le résultat ressemblera au contour de l'objet.

```
#Morphological Gradient
gradient = cv2.morphologyEx(a1, cv2.MORPH_GRADIENT, kernel)
plt.imshow(gradient,cmap='gray')
```

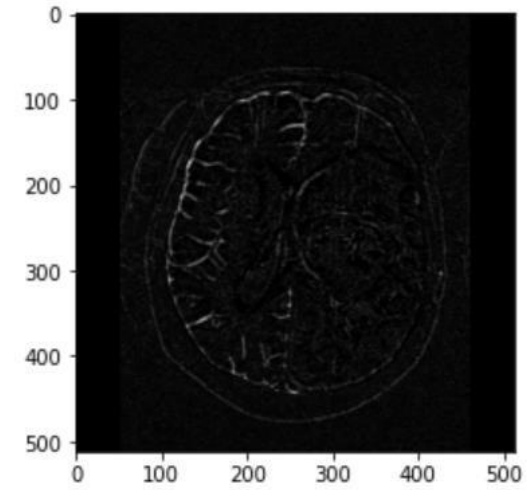


2.7.6 Top Hat

C'est la différence entre l'image d'entrée et l'ouverture de l'image.

```
#Top Hat
tophat = cv2.morphologyEx(a1, cv2.MORPH_TOPHAT, kernel)
plt.imshow(tophat,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3a91e35710>
```

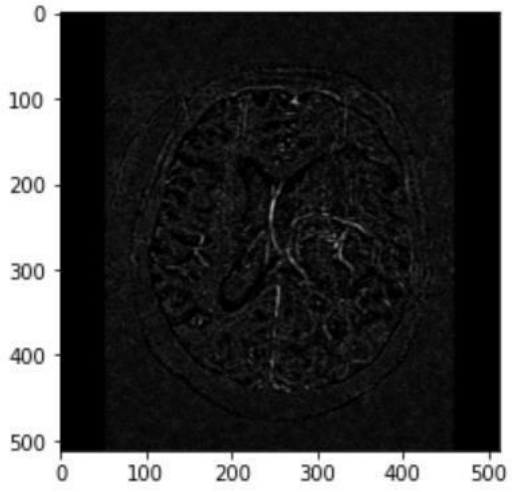


2.7.7 Black Hat

C'est la différence entre la fermeture de l'image d'entrée et de l'image d'entrée.

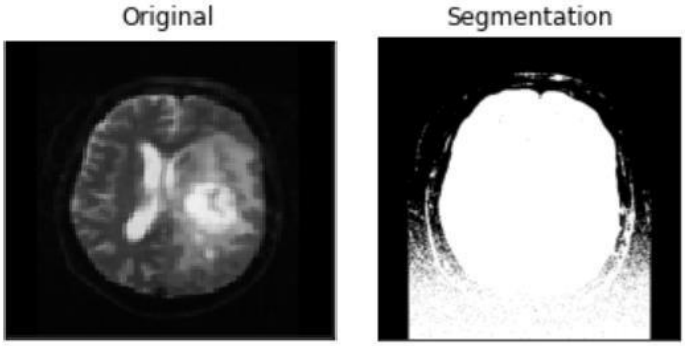
```
[ ]  blackhat = cv2.morphologyEx(a1, cv2.MORPH_BLACKHAT, kernel)
     plt.imshow(blackhat,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f3a91d90810>
```



2.8 Segmentation de l'image

```
#gray = rgb2gray(a1)
plt.imshow(gray, cmap='gray')
gray.shape
gray_r = gray.reshape(gray.shape[0] * gray.shape[1])
for i in range(gray_r.shape[0]):
        if gray_r[i] > gray_r.mean():
            gray_r[i] = 1
        else:
            gray_r[i] = 0
gray = gray_r.reshape(gray.shape[0], gray.shape[1])

plt.subplot(121), plt.imshow(a1, cmap='gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(gray, cmap='gray')
plt.title('Segmentation'), plt.xticks([]), plt.yticks([])
plt.show()
```



2- Classification et interprétation

D'abord qu'est-ce qu'un AVC ?

L'AVC est une urgence médicale. Un accident vasculaire cérébral survient lorsque le flux sanguin vers une partie de votre cerveau est interrompu ou réduit, empêchant les tissus cérébraux d'obtenir de l'oxygène et des nutriments. Les cellules cérébrales commencent à mourir en quelques minutes.

L'AVC est une condition médicale qui peut entraîner la mort d'une personne. C'est une maladie grave et si elle est traitée à temps, nous pouvons sauver sa vie et bien la traiter. Il peut y avoir n nombre de facteurs qui peuvent conduire à des accidents vasculaires cérébraux et dans ce rapport, nous essaierons d'en analyser quelques-uns. Notre ensemble de données comporte 11 variables et 5110 observations.

Importation de bibliothèques :

Pour accomplir n'importe quelle tâche, nous avons besoin d'outils, et nous avons beaucoup d'outils en python. Commençons par importer les bibliothèques requises.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score,classification_report,precision_score,recall_score
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
```

Lecture de CSV

Lire les fichiers CSV, qui contiennent nos données. Avec l'aide de ce CSV, nous essaierons de comprendre le modèle et de créer notre modèle de prédiction.

```
[ ]  data=pd.read_csv('healthcare-dataset-stroke-data.csv')
     data.head(10)
     ## Displaying top 10 rows
     data.info()
     ## Showing information about datase
     data.describe()
     ## Showing data's statistical features
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknown	1
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1

Info :

```
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
```

Description:

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

IDENTIFIANT

L'identification n'est rien d'autre qu'un numéro unique attribué à chaque patient pour en garder une trace et les rendre uniques. Il n'y a pas besoin de pièce d'identité, c'est complètement inutile alors supprimons-le.

```
[ ]  data.drop("id",inplace=True,axis=1)
```

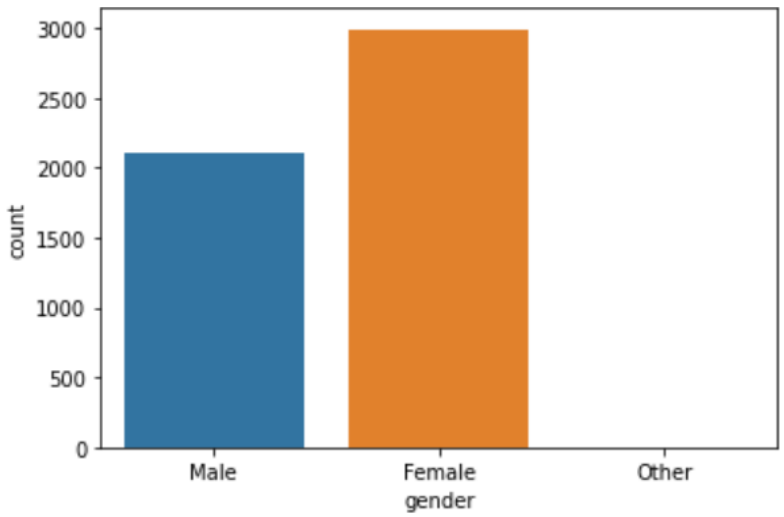
Le sexe

Cet attribut indique le sexe du patient. Voyons comment le sexe affecte et compare le taux d'AVC en fonction du sexe.

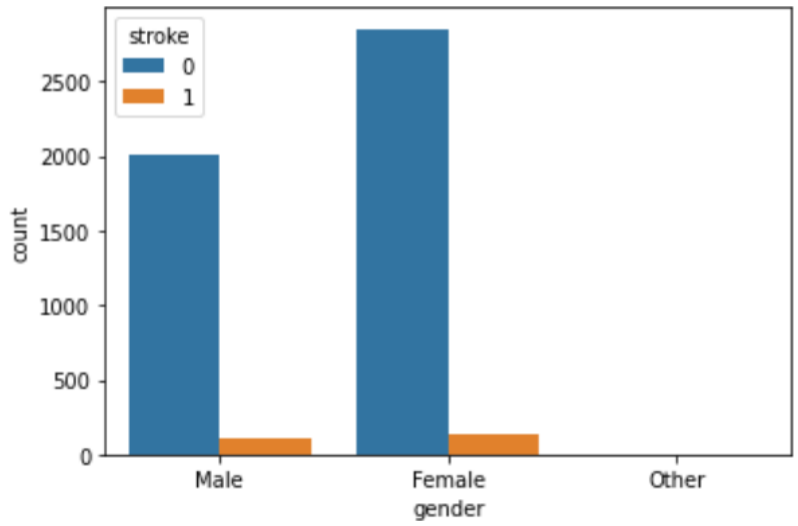
```
[ ]  print('Unique values\n',data['gender'].unique())
     print('Value Counts\n',data['gender'].value_counts())
     # Above codes will help to give us information about it's unique values and count of each value.

     sns.countplot(data=data,x='gender')
     # Helps to plot a count plot which will help us to see count of values in each unique category.
     sns.countplot(data=data,x='gender',hue='stroke')
     # This plot will help to analyze how gender will affect chances of stroke.
```

```
Unique values
['Male' 'Female' 'Other']
Value Counts
Female    2994
Male      2115
Other        1
```

Gender Plot:



Gender with Stroke:



Observation:

On dirait que l'ensemble de données est déséquilibré. Quoi qu'il en soit, comme nous le pouvons, il n'y a pas beaucoup de différence entre le taux d'AVC concernant le sexe

Âge

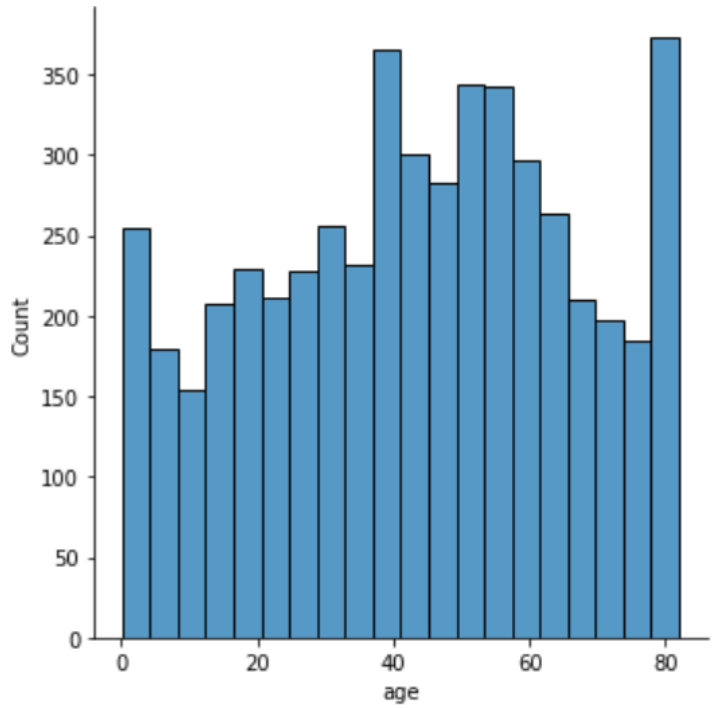
Eh bien ici, l'âge n'est tout simplement pas un nombre, c'est l'un des facteurs importants ou, comme on peut le dire, c'est un facteur très crucial. Analysons nos données et voyons quel impact réel elles ont.

```
[ ]  data['age'].nunique()
     # Returns number of unique values in this attribute
     sns.displot(data['age'])
     # This will plot a distribution plot of variable age
     plt.figure(figsize=(15,7))
     sns.boxplot(data=data,x='stroke',y='age')
     # Above code will plot a boxplot of variable age with respect of target attribute stroke
```

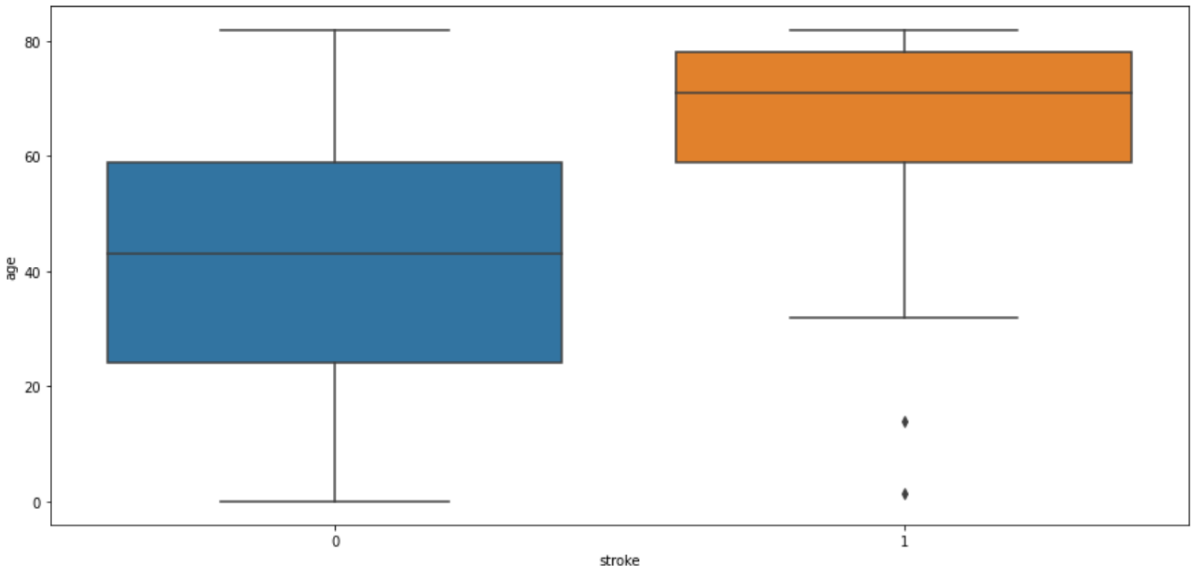
Nombre de valeurs uniques :

```
104
```

Distribution Plot:



Âge et AVC :



Observation:

Les personnes âgées de plus de 60 ans ont tendance à faire un AVC. Certaines valeurs aberrantes peuvent être considérées comme des personnes de moins de 20 ans ayant un AVC, il est possible que ce soit des données valides car l'AVC dépend également de nos habitudes alimentaires et de vie. Une autre observation est que les personnes n'ayant pas d'AVC sont également des personnes âgées de plus de 60 ans.

Hypertension

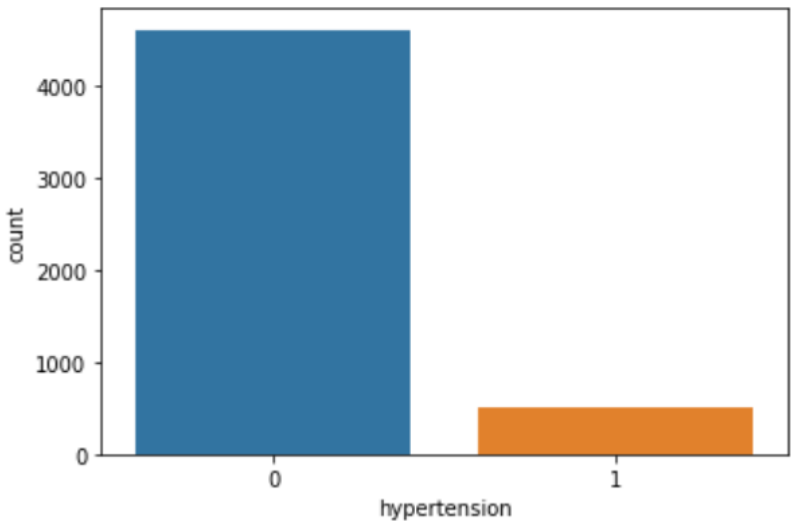
L'hypertension est une condition lorsqu'une personne souffre d'hypertension artérielle. L'hypertension peut entraîner un accident vasculaire cérébral. Voyons comment ça se passe.

```
[ ]  data['age'].nunique()
     # Returns number of unique values in this attribute
     sns.displot(data['age'])
     # This will plot a distribution plot of variable age
     plt.figure(figsize=(15,7))
     sns.boxplot(data=data,x='stroke',y='age')
     # Above code will plot a boxplot of variable age with respect of target attribute stroke
```

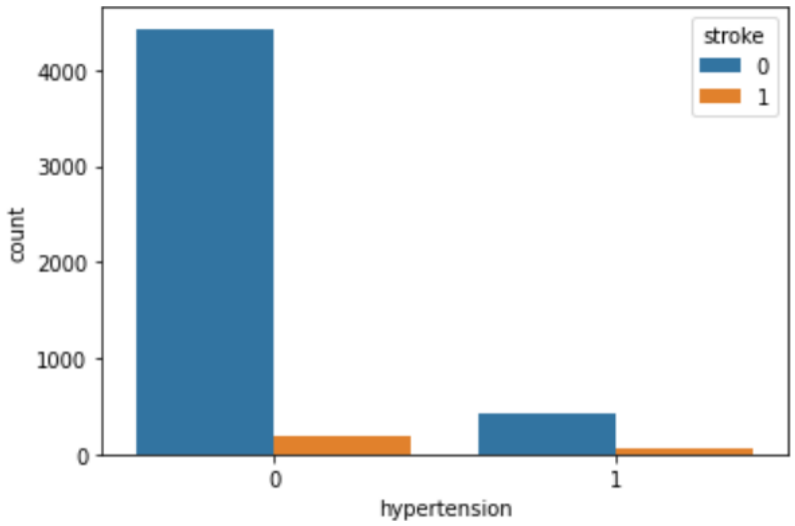
Valeurs uniques et nombre de valeurs :

```
Value Count [0 1]
Value Counts
0 4612
1 498
```

Count Plot:



Hypertension et AVC :



Observation:

l'hypertension est rare chez les jeunes et fréquente chez les personnes âgées. L'hypertension peut provoquer un accident vasculaire cérébral. Sur la base de nos données, l'image n'est pas si claire pour l'hypertension. Il a assez peu de données sur les patients souffrant d'hypertension.

Jamais marié

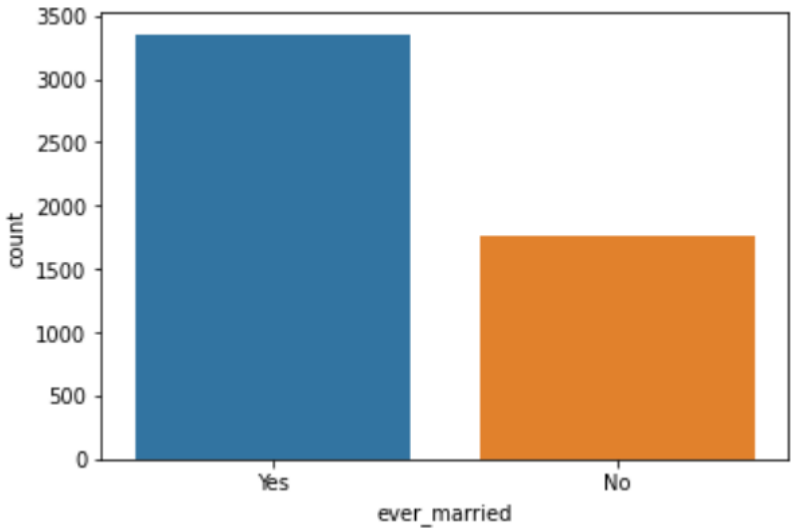
Cet attribut nous dira si oui ou non le patient a déjà été marié. Voyons comment cela affectera les chances d'avoir un accident vasculaire cérébral.

```
print('Unique Values\n',data['ever_married'].unique())
print('Value Counts\n',data['ever_married'].value_counts())
# Above code will show us number unique values of attribute and its count
sns.countplot(data=data,x='ever_married')
# Counter plot of ever married attribute
sns.countplot(data=data,x='ever_married',hue='stroke')
# Ever married with respect of stroke
```

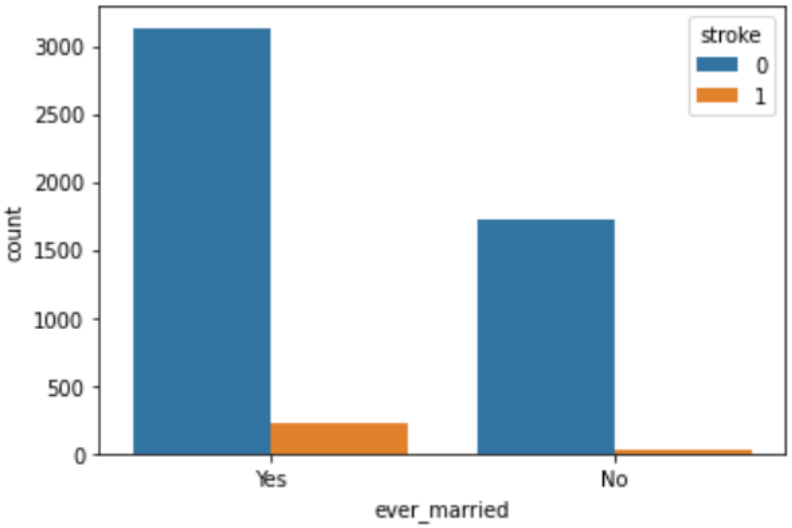
Valeurs et nombre uniques :

```
Unique Values
['Yes' 'No']
Value Counts
Yes    3353
No     1757
```

Count Plot:



Déjà marié avec un AVC :



Observation:

Les personnes mariées ont un taux d'AVC plus élevé.

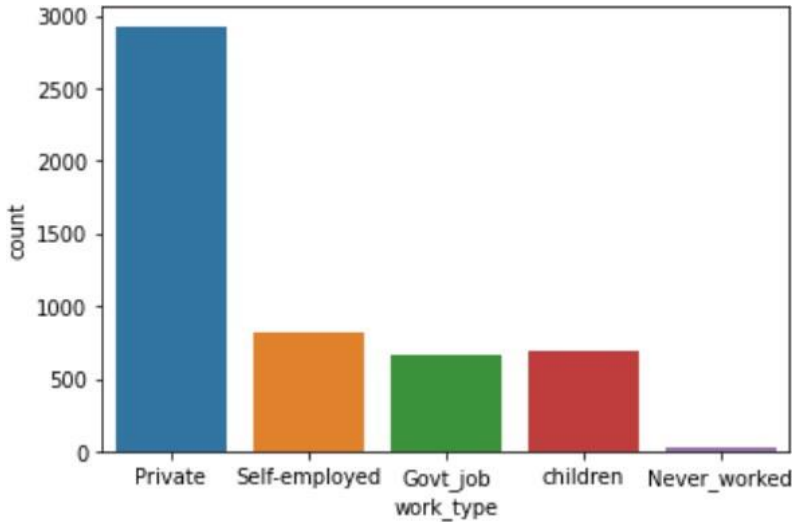
Type de travail

Cet attribut contient des données sur le type de travail effectué par le patient. Différents types de travail ont différents types de problèmes et de défis qui peuvent être la cause possible de l'excitation, du frisson, du stress, etc. Le stress n'est jamais bon pour la santé, voyons comment cette variable peut affecter les chances d'avoir un AVC.

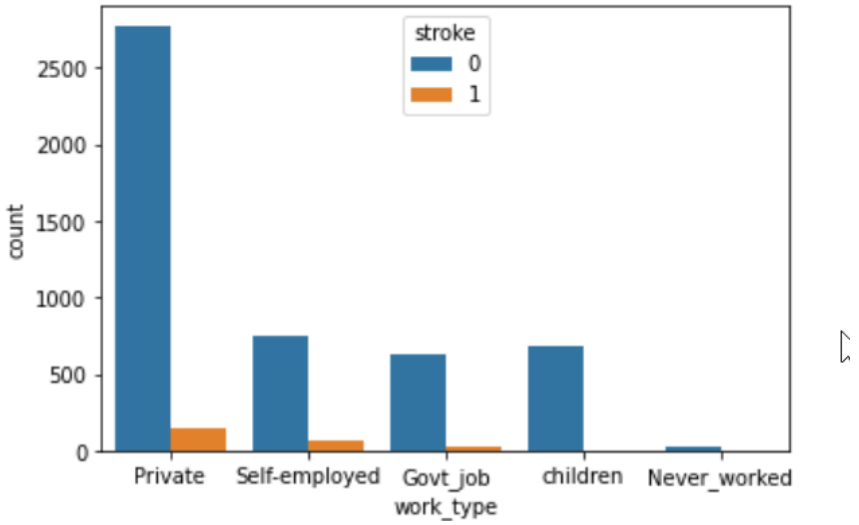
```
[ ]  print('Unique Value\n',data['work_type'].unique())
     print('Value Counts\n',data['work_type'].value_counts())
     # Above code will return unique values of attributes and its count
     sns.countplot(data=data,x='work_type')
     # Above code will create a count plot
     sns.countplot(data=data,x='work_type',hue='stroke')
     # Above code will create a count plot with respect to stroke
```

Valeurs et nombre uniques :

Count Plot:



Type de travail et AVC



Observation:

Les personnes travaillant dans le secteur privé ont un risque plus élevé de subir un AVC. Et les personnes qui n'ont jamais travaillé ont un taux d'AVC très inférieur.

Type de résidence

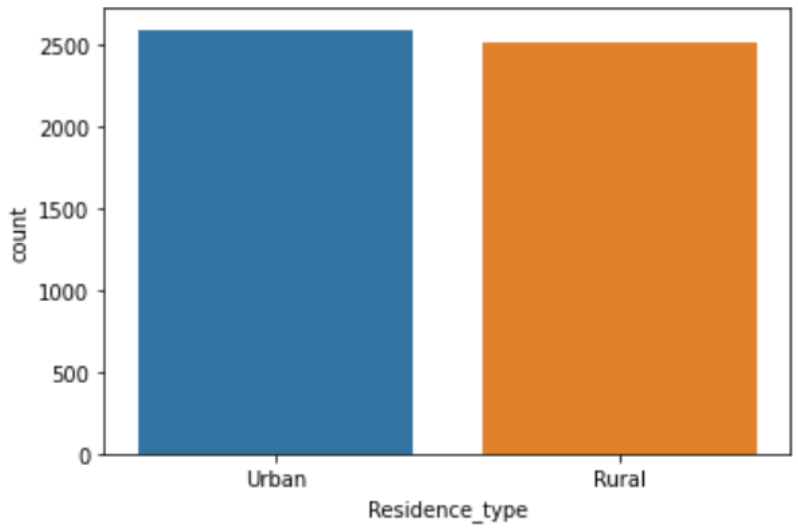
Cet attribut nous indique si le type de résidence du patient est. Urbain ou Rural.

```
[ ]  print('Unique Values\n',data['Residence_type'].unique())
     print("Value Counts\n",data['Residence_type'].value_counts())
     # Above code will return unique values of variable and its count
     sns.countplot(data=data,x='Residence_type')
     # This will create a counter plot
     sns.countplot(data=data,x='Residence_type',hue='stroke')
     # Residence Type with respect to stroke
```

Valeurs et nombre uniques :

```
Unique Values
['Urban' 'Rural']
Value Counts
Urban    2596
Rural    2514
```

Counter Plot:



Niveau moyen de glucose

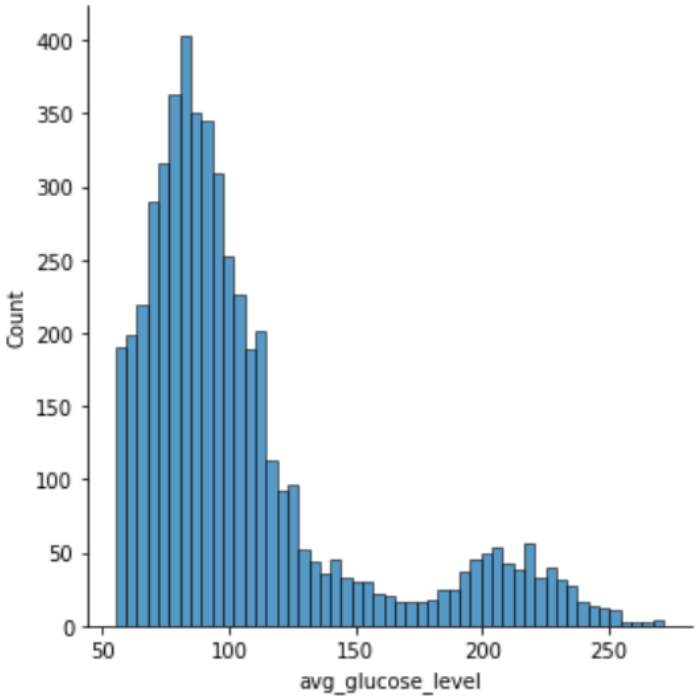
Indique le niveau moyen de glucose dans le corps du patient. Voyons si cela affecte les chances d'avoir un AVC

```
[ ]  data['avg_glucose_level'].nunique()
     # Number of unique values
     sns.displot(data['avg_glucose_level'])
     # Distribution of avg_glucose_level
     sns.boxplot(data=data,x='stroke',y='avg_glucose_level')
     # Avg_glucose_level and Stroke
```

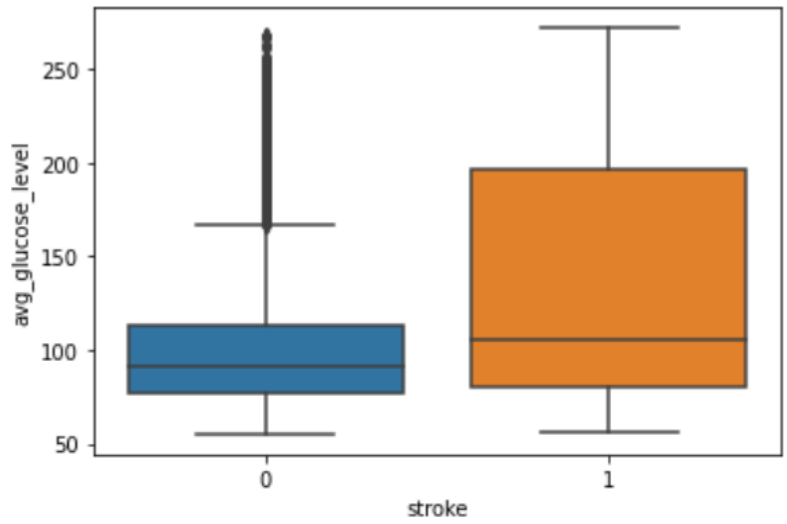
Unique Values and Count:

```
3979
```

Distribution Plot:



Glycémie et AVC :



IMC

L'indice de masse corporelle est une mesure de la graisse corporelle basée sur la taille et le poids qui s'applique aux hommes et aux femmes adultes. Voyons comment cela affecte les chances d'avoir un AVC.

```
[ ]  data['bmi'].isna().sum()
     # Returns number null values
     data['bmi'].fillna(data['bmi'].mean(),inplace=True)
     # Filling null values with average value
     data['bmi'].nunique()
     # Returns number of unique values in that attribute
     sns.displot(data['bmi'])
     # Distribution of bmi
     sns.boxplot(data=data,x='stroke',y='bmi')
     # BMI with respect to Stroke
```

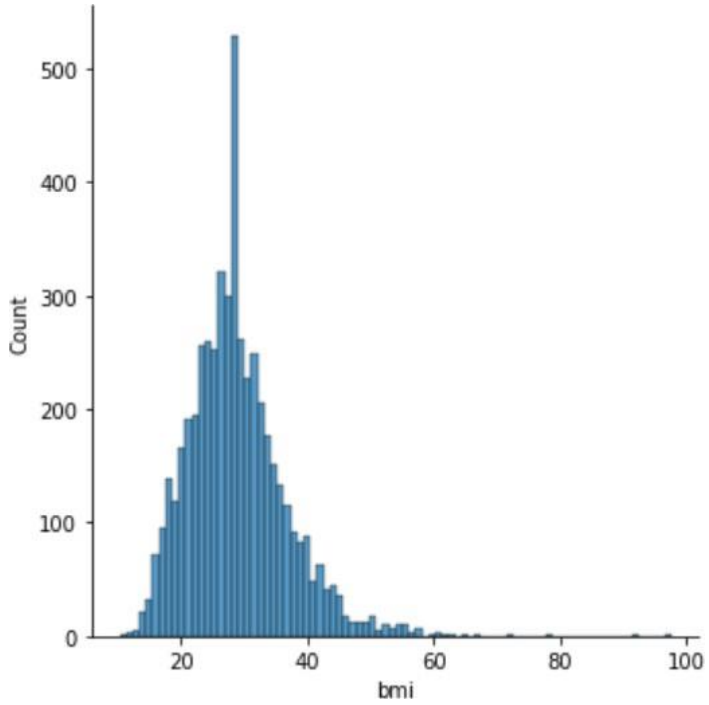
Null Values:

291

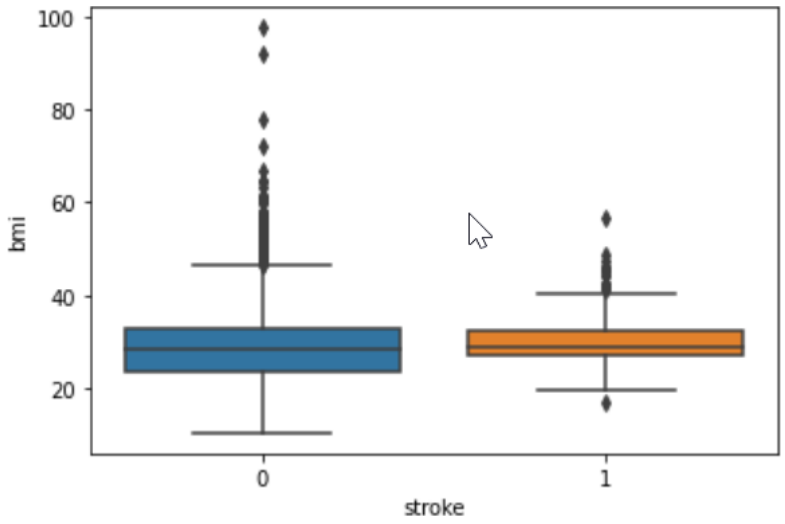
Unique Values and Counts:

419

Distribution Plot:



IMC ET AVC



Observation:

Il n'y a en tant que tel aucune observation importante sur la façon dont l'IMC affecte les chances d'avoir un accident vasculaire cérébral.

Statut de fumeur

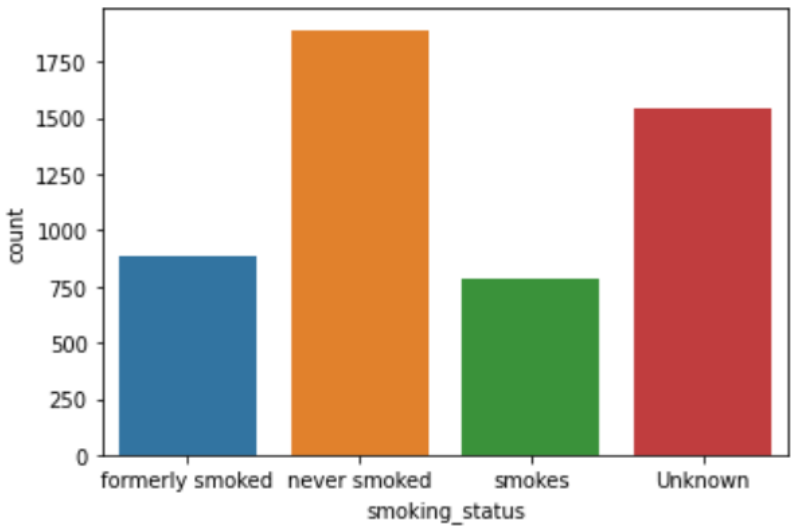
Ces attributs nous indiquent si le patient fume ou non. Fumer est nocif pour la santé et peut provoquer des maladies cardiaques. Voyons comment cela se passe dans le cas de nos données

```
[ ]  print('Unique Values\n',data['smoking_status'].unique())
     print('Value Counts\n',data['smoking_status'].value_counts())
     # Returns unique values and its count
     sns.countplot(data=data,x='smoking_status')
     # Count plot of smoking status
     sns.countplot(data=data,x='smoking_status',hue='stroke')
     # Smoking Status with respect to Stroke
```

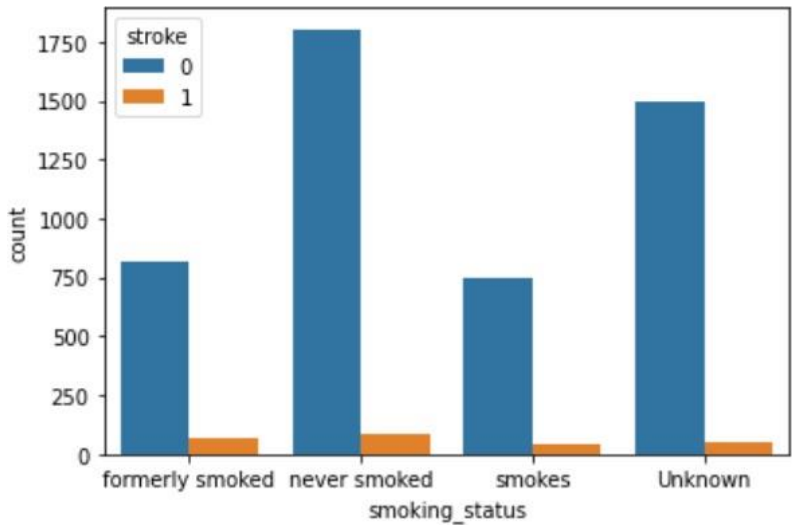
Unique Values and Count:

```
Unique Values
['formerly smoked' 'never smoked' 'smokes' 'Unknown']
Value Counts
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
```

Count Plot:



Smoke and Stroke:



Observation:

Selon ces graphiques, nous pouvons voir qu'il n'y a pas beaucoup de différence dans les risques d'AVC, quel que soit le statut de fumeur.

AVC

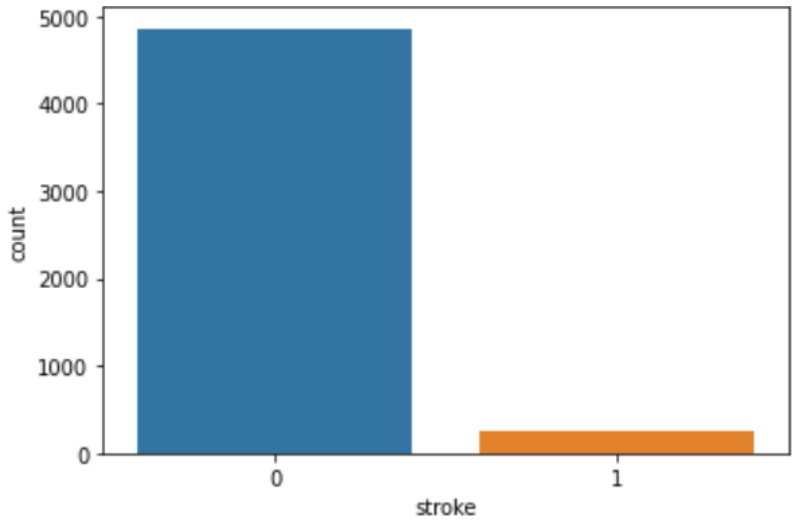
Notre variable cible. Il nous indique si les patients ont des risques d'AVC.

```
[ ]  print('Unique Value\n',data['stroke'].unique())
     print('Value Counts\n',data['stroke'].value_counts())
     # Returns Unique Value and its count
     sns.countplot(data=data,x='stroke')
     # Count Plot of Stroke
```

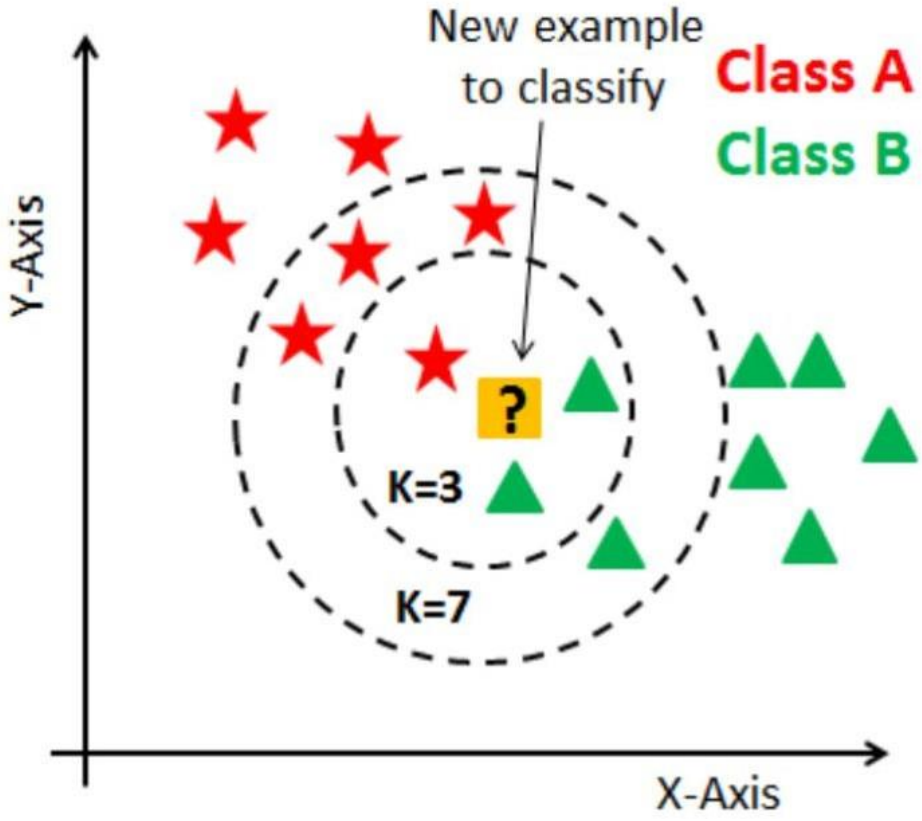
Unique Values and Count:

```
Unique Value
[1 0]
Value Counts
0    4861
1     249
```

Count Plot:



K-nearest-neighbors-algorithm:



K Nearest Neighbor est un algorithme simple qui stocke tous les cas disponibles et classe les nouvelles données ou cas en fonction d'une mesure de similarité. Il est principalement utilisé pour classer un point de données en fonction de la façon dont ses voisins sont classés.

```
# Estimate Knn model and report the outcome :
from sklearn.neighbors import KNeighborsClassifier
```

```
# Create the model :
knn = KNeighborsClassifier(n_neighbors=3)
```

```
# Train (fit) the model :
knn = knn.fit(x_train, y_train)
```

```
# Make a prediction at the first 5 rows regrads (k = 3) till we dertmine the optimum value of k
y_pred = knn.predict(x_test)
y_pred[:5]
```

```
array([0, 0, 0, 0, 0])
```

Validation et précision du modèle :

```
from sklearn.metrics import accuracy_score, classification_report, f1_score
```

```
print(classification_report(y_test, y_pred))
print('Accuracy Score:', round(accuracy_score(y_test, y_pred,2)))
print("F1_score: ", round(f1_score(y_test, y_pred,2)))
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       951
           1       0.10      0.02      0.03        49

    accuracy                           0.94      1000
   macro avg       0.53      0.51      0.50      1000
weighted avg       0.91      0.94      0.92      1000

Accuracy Score: 1
F1_score:  0
```

Appliquer le classificateur d'arbres de décision :

L'apprentissage par arbre de décision ou l'induction d'arbres de décision est l'une des approches de modélisation prédictive utilisées dans les statistiques, l'exploration de données et l'apprentissage automatique. Il utilise un arbre de décision (comme modèle prédictif) pour passer des observations sur un élément (représentées dans les branches) aux

conclusions sur la valeur cible de l'élément (représentées dans les feuilles). Les modèles d'arbres où la variable cible peut prendre un ensemble discret de valeurs sont appelés arbres de classification ; dans ces arborescences, les feuilles représentent les étiquettes de classe et les branches représentent les conjonctions d'entités qui conduisent à ces étiquettes de classe. Les arbres de décision où la variable cible peut prendre des valeurs continues (généralement des nombres réels) sont appelés arbres de régression. Les arbres de décision font partie des algorithmes d'apprentissage automatique les plus populaires en raison de leur intelligibilité et de leur simplicité.

1/ Ajuster un classificateur d'arbre de décision sans AUCUNE limite définie sur les éléments de profondeur maximale et les feuilles.

2/ Déterminer combien de nœuds sont présents et quelle est la profondeur de cet arbre (très grand) ?.

3/ Utiliser cet arbre pour mesurer l'erreur de prédiction dans l'ensemble de données d'apprentissage et de test.

```
from sklearn.tree import DecisionTreeClassifier
```

```
# Create the model
dt = DecisionTreeClassifier(random_state=42)
```

```
# Fit the model
dt = dt.fit(x_train, y_train)
```

```
# Determine the number of nodes and maximum depth:
dt.tree_.node_count, dt.tree_.max_depth
```

```
(657, 18)
```

Identifiez les métriques d'erreur :

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
def measure_error(y_true, y_pred, label):
  return pd.Series({'accuracy': accuracy_score(y_true, y_pred),
                    'precision': precision_score(y_true, y_pred),
                    'recall': recall_score(y_true, y_pred),
                    'f1': f1_score(y_true, y_pred)},
                   name=label)
```

```
# This step may lead to overfitting because we did not prune the tree:
y_train_pred = dt.predict(x_train)
```

```
y_test_pred = dt.predict(x_test)
```

```
y_test_pred[:5]
```

```
array([0, 0, 0, 0, 0])
```

Vérifiez la précision du modèle et la mesure de l'erreur :

```
train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred,'test')],axis=1)
```

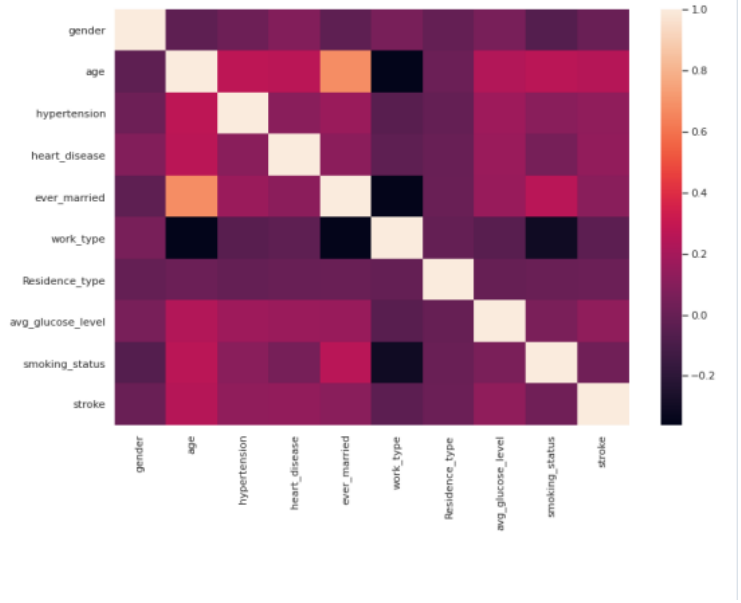
```
train_test_full_error
```

	train	test
accuracy	1.0	0.915000
precision	1.0	0.166667
recall	1.0	0.183673
f1	1.0	0.174757

CORRÉLATION

```
# Create a heatmap for checking variables correlations :
fig, ax = plt.subplots(figsize=(12, 8))
sns.heatmap(df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4ad9b52510>
```




```
import warnings
```

```
warnings.filterwarnings("ignore", category= UserWarning)
warnings.filterwarnings("ignore", category= RuntimeWarning)
```

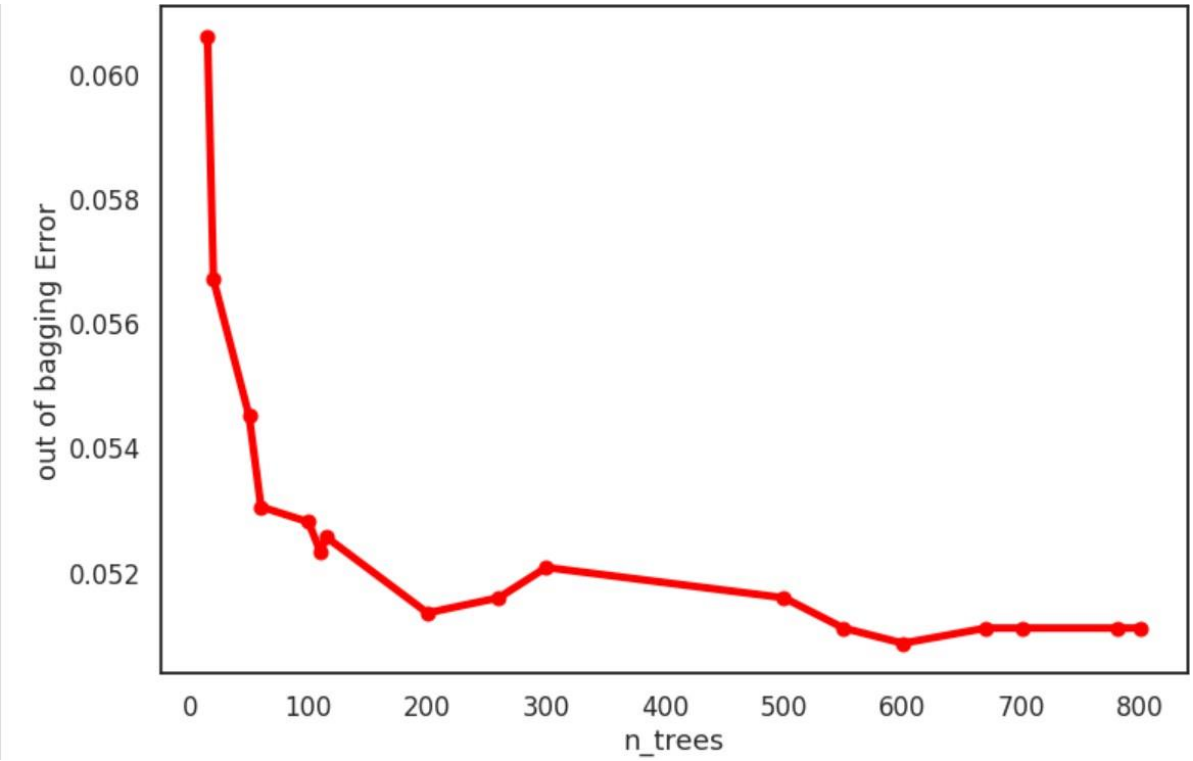
```
from sklearn.ensemble import RandomForestClassifier
```

```
RF = RandomForestClassifier(oob_score=True,
                            random_state=42,
                            warm_start=True,
                            n_jobs = -1)

oob_list = list()
for n_trees in [15, 20, 50 , 60, 100, 110, 115, 200, 260, 300, 500, 550, 600, 670, 700, 780, 800]:
  RF.set_params(n_estimators=n_trees)

  # Fit the model :
  RF.fit(x_train, y_train)

  # Get the oob Erros:
  oob_error = 1 - RF.oob_score_
  # Score it :
  oob_list.append(pd.Series({'n_trees': n_trees, 'OOb':oob_error}))

rf_oob_df = pd.concat(oob_list, axis=1).T.set_index('n_trees')
rf_oob_df
```

	OOb
n_trees	
15.0	0.060584
20.0	0.056691
50.0	0.054501
60.0	0.053041
100.0	0.052798
110.0	0.052311
115.0	0.052555
200.0	0.051338
260.0	0.051582
300.0	0.052068
500.0	0.051582
550.0	0.051095
600.0	0.050852
670.0	0.051095
700.0	0.051095
780.0	0.051095
800.0	0.051095

```
sns.set_context('talk')
sns.set_style('white')
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(12, 8),linewidth=5, color='red')
ax.set(ylabel='out of bagging Error')
```

```
[Text(0, 0.5, 'out of bagging Error')]
```



dans ce rapport , nous avons vu certains des facteurs pouvant entraîner des accidents vasculaires cérébraux. Où l'âge était fortement corrélé, suivi de l'hypertension, des maladies cardiaques, de la glycémie moyenne et du fait d'avoir déjà été marié.

K –nearest-neighbors-algorithm était un chevalier qui a bien performé. Il y a des valeurs aberrantes dans certaines variables, raison pour laquelle on la gardé tel quel parce que ces choses dépendent soit d'autres facteurs et il y a des possibilités d'avoir ce genre d'enregistrements. Par exemple, l'IMC peut être élevé et toujours pas d'AVC car une personne est jeune ou n'a pas de maladie cardiaque.

3- CONCLUSION

L'imagerie médicale joue un rôle crucial dans les soins de santé à tous les niveaux importants. En plus de fournir des outils clés pour l'analyse clinique et le diagnostic, il est tout aussi important pour le traitement lui-même. Sans elle, les médecins auraient à recourir à des méthodes de diagnostic invasives beaucoup plus souvent. Le suivi des progrès au cours du traitement serait beaucoup plus difficile, voire impossible, et le traitement des patients en soins de suivi ne comprendrait pas de bases de données utiles à titre de référence.

La technologie médicale s'améliore et se diversifie constamment pour s'adapter aux exigences complexes des systèmes d'imagerie médicale.