## BASKARAN

## Amalan

## N°12001105


## 19/11/2023

# ---------- TP 4 : Segmentation d'images microscopiques ----------

Entrée [1]:

```
import numpy as np
import pandas as pd # Toutes les libraries utlilsé au cours de ce TP.
import cv2
from skimage.filters import roberts,sobel,scharr,prewitt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from scipy import ndimage as nd
import pickle
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

## A) Prise en main des images

Train_images contiennt des images microscopique en noir et blancs. Train-masks contiennt des même images mais segmenté.

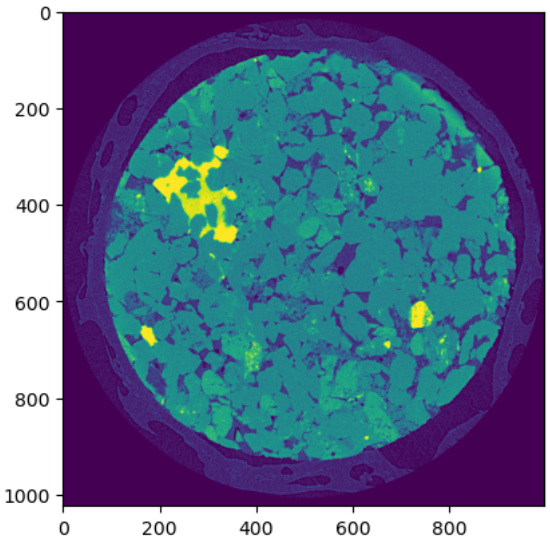
## B) Extraction des attributs et preparation du jeu de données

Entrée [2]:

```
Img = cv2.imread("C:/Users/Latitude 7280/Desktop/TP ML/TP4/Train_images/Sandstone_Versa0000.tif") # importation
Img = cv2.cvtColor(Img, cv2.COLOR_BGR2GRAY) # en gris
Img2 =Img.reshape(-1) # 1D
plt.imshow(Img)
```

Out[2]:

```
<matplotlib.image.AxesImage at 0x253bb68f210>
```



Entrée [3]:

```
df = pd.DataFrame() # creation de dataframe
df['Original Imag'] = Img2 #ajouter les données de img2 à la colonne 'original imag'
print(df.head)
```

```
<bound method NDFrame.head of         Original Imag
0                   0
1                   0
2                   0
3                   0
4                   0
...               ...
1019899             0
1019900             0
1019901             0
1019902             0
1019903             0

[1019904 rows x 1 columns]>
```

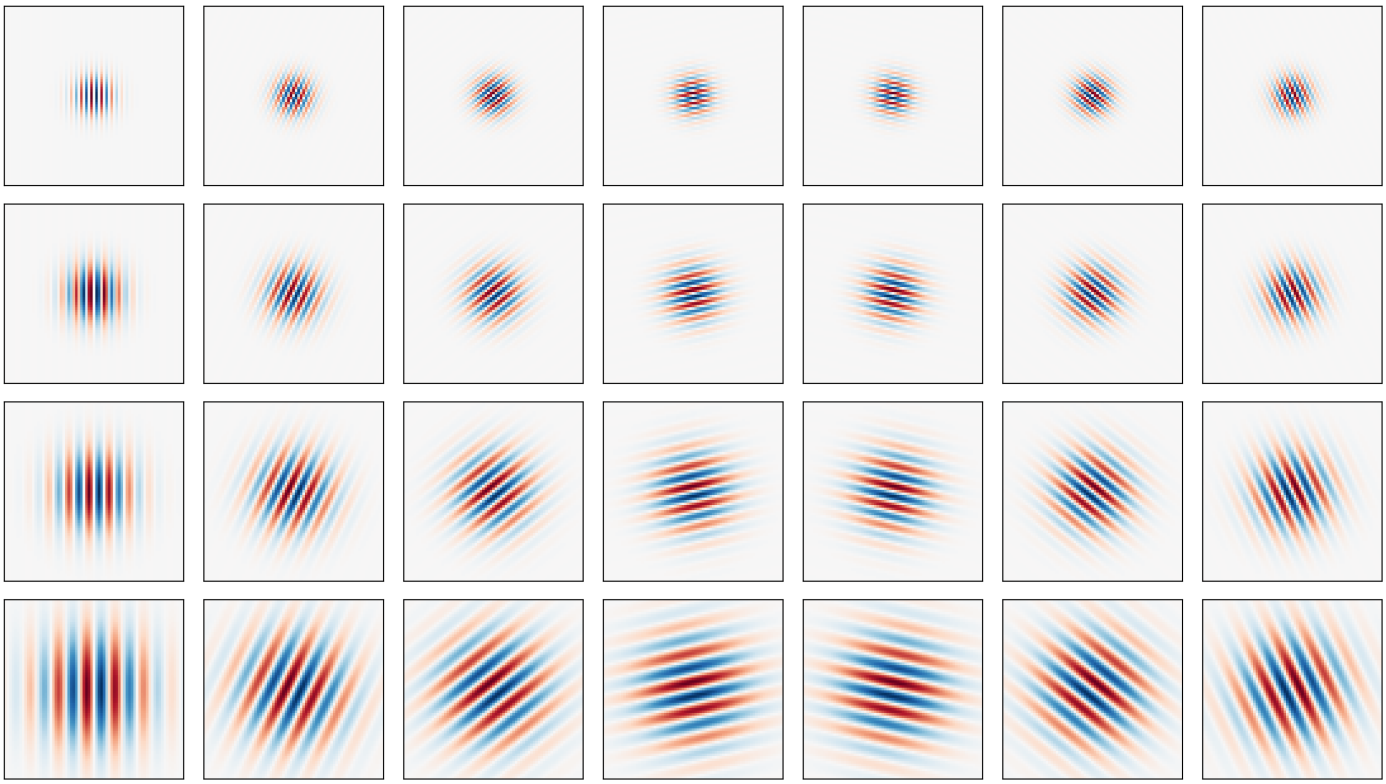
## Fonction et Filtre de Gabor

$$G(x,y,\theta,f)=e^{-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2}\right)}\cos(2\pi f x_\theta)$$

avec $x_\theta = x\cos\theta + y\sin\theta$

et $y_\theta = y\cos\theta - x\sin\theta$

où $\theta$ est l'orientation de la sinusoïde, $f$ sa fréquence et $\sigma_x$ (*respectivement* $\sigma_y$) l'écart-type de la gaussienne selon l'axe des abscisses (*resp.* des ordonnées).



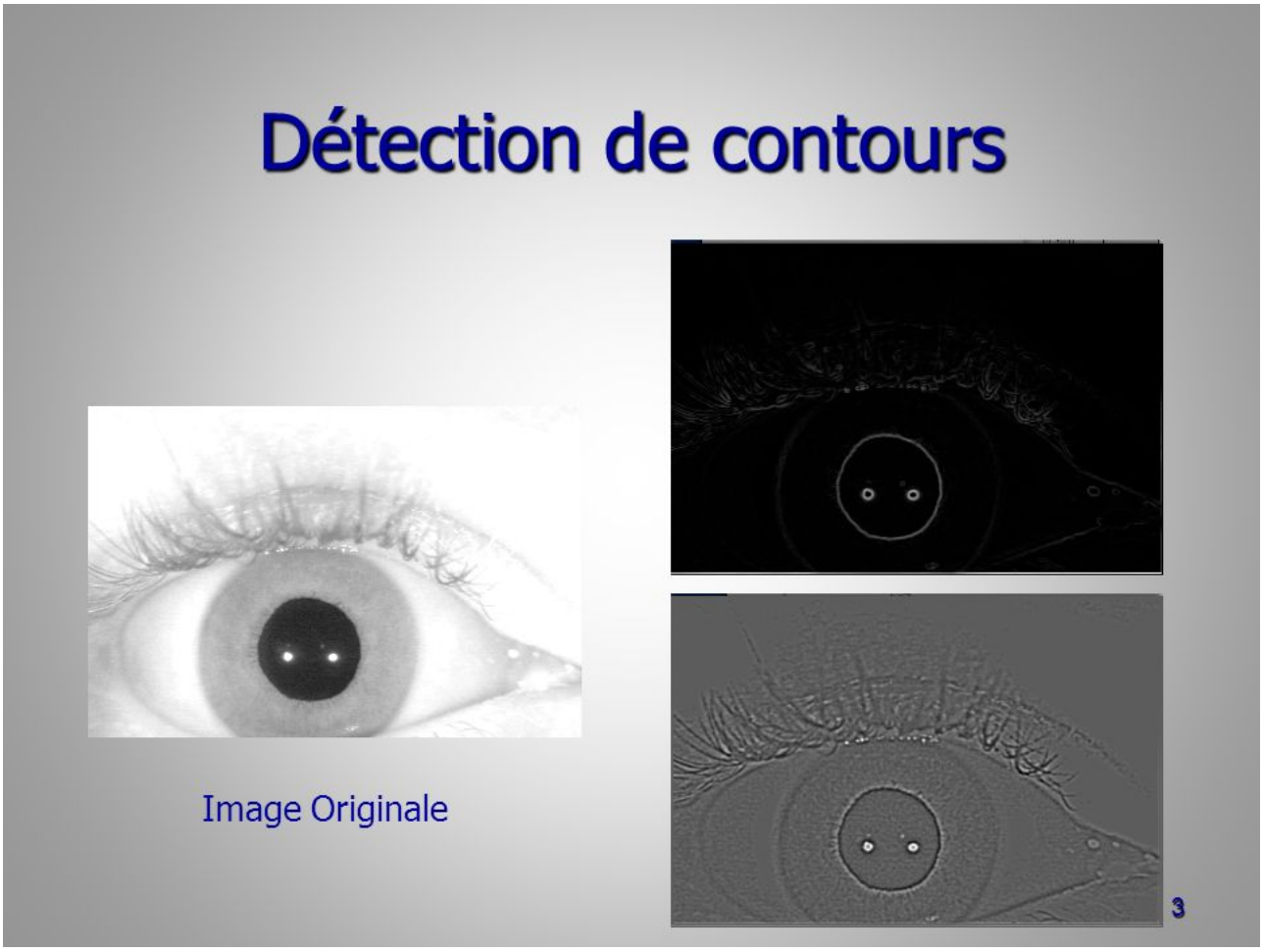
Le filtre de Gabor est un outil puissant pour l'analyse d'images, car il permet de détecter des caractéristiques locales dans différentes orientations et fréquences. On peut contrôler différents paramètres avec le filtre de Gabor, comme la direction, la fréquence, la taille et la forme du noyau, ou encore le rapport d'aspect. Ces paramètres influencent la sensibilité et la sélectivité du filtre, ainsi que sa capacité à extraire des informations pertinentes de l'image.

Entrée [4]:

```
# générer getGaborKernel pour différer en paramètres.
num = 1
kernels = []
for theta in range(2): # orientation du filtre en degrés
    theta = theta / 4. * np.pi # conv en radian
    for sigma in (1, 3): #L'écart-type de l'enveloppe gaussienne
        for lamda in np.arange(0, np.pi, np.pi / 4): #la longueur d'onde du facteur sinusoïdal
            for gamma in (0.05, 0.5): #le rapport d'aspect spatial du filtre
                gabor_label = 'Gabor' + str(num) #string pour affichage
                print(gabor_label)
                ksize=9 #c'est la taille du noyau de filtre
                kernel = cv2.getGaborKernel((ksize, ksize), sigma, theta,
                lamda, gamma, 0, ktype=cv2.CV_32F) #Créer le noyau de filtre de Gabor
                kernels.append(kernel) # Ajoute dans une liste.
                fimg = cv2.filter2D(Img2, cv2.CV_8UC3, kernel) # Appliquer le filtre de Gabor à l'image
                filtered_img = fimg.reshape(-1) # 1D
                df[gabor_label] = filtered_img # ajouter en tant qu'une nouvelle colonne dans la dataframe df comme gabor_label.
                print(gabor_label, ': theta=', theta, ': sigma=', sigma, ': lamda=',
                lamda, ': gamma=', gamma) # print les paramètres.
                num += 1 # incrementation de n.
```

```
Gabor1
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.0 : gamma= 0.05
Gabor2
Gabor2 : theta= 0.0 : sigma= 1 : lamda= 0.0 : gamma= 0.5
Gabor3
Gabor3 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.05
Gabor4
Gabor4 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor5
Gabor5 : theta= 0.0 : sigma= 1 : lamda= 1.5707963267948966 : gamma= 0.05
Gabor6
Gabor6 : theta= 0.0 : sigma= 1 : lamda= 1.5707963267948966 : gamma= 0.5
Gabor7
Gabor7 : theta= 0.0 : sigma= 1 : lamda= 2.356194490192345 : gamma= 0.05
Gabor8
Gabor8 : theta= 0.0 : sigma= 1 : lamda= 2.356194490192345 : gamma= 0.5
Gabor9
Gabor9 : theta= 0.0 : sigma= 3 : lamda= 0.0 : gamma= 0.05
Gabor10
Gabor10 : theta= 0.0 : sigma= 3 : lamda= 0.0 : gamma= 0.5
Gabor11
Gabor11 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.05
Gabor12
Gabor12 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor13
Gabor13 : theta= 0.0 : sigma= 3 : lamda= 1.5707963267948966 : gamma= 0.05
Gabor14
Gabor14 : theta= 0.0 : sigma= 3 : lamda= 1.5707963267948966 : gamma= 0.5
Gabor15
Gabor15 : theta= 0.0 : sigma= 3 : lamda= 2.356194490192345 : gamma= 0.05
Gabor16
Gabor16 : theta= 0.0 : sigma= 3 : lamda= 2.356194490192345 : gamma= 0.5
Gabor17
Gabor17 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.0 : gamma= 0.05
Gabor18
Gabor18 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.0 : gamma= 0.5
Gabor19
Gabor19 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.05
Gabor20
Gabor20 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor21
Gabor21 : theta= 0.7853981633974483 : sigma= 1 : lamda= 1.5707963267948966 : gamma= 0.05
Gabor22
Gabor22 : theta= 0.7853981633974483 : sigma= 1 : lamda= 1.5707963267948966 : gamma= 0.5
Gabor23
Gabor23 : theta= 0.7853981633974483 : sigma= 1 : lamda= 2.356194490192345 : gamma= 0.05
Gabor24
Gabor24 : theta= 0.7853981633974483 : sigma= 1 : lamda= 2.356194490192345 : gamma= 0.5
Gabor25
Gabor25 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.0 : gamma= 0.05
Gabor26
Gabor26 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.0 : gamma= 0.5
Gabor27
Gabor27 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.05
Gabor28
Gabor28 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor29
Gabor29 : theta= 0.7853981633974483 : sigma= 3 : lamda= 1.5707963267948966 : gamma= 0.05
Gabor30
Gabor30 : theta= 0.7853981633974483 : sigma= 3 : lamda= 1.5707963267948966 : gamma= 0.5
Gabor31
Gabor31 : theta= 0.7853981633974483 : sigma= 3 : lamda= 2.356194490192345 : gamma= 0.05
Gabor32
Gabor32 : theta= 0.7853981633974483 : sigma= 3 : lamda= 2.356194490192345 : gamma= 0.5
```

### Detection des contours




La détection de contours est une étape importante dans la segmentation d'une image, car elle permet de séparer les régions d'intérêt de l'arrière-plan.

**Dans cette partie nous allons appliquer à l'image divers algorithmes de detection de contours comme canny,sobel,roberts,prewitt, etc...**

Entrée [5]:

```
# plusieurs operations de détection de contours
edges = cv2.Canny(Img, 100,200) #appliquer la detection de contrours canny
edges1 = edges.reshape(-1) # convert 1D
df['Canny Edge'] = edges1 #Ajout d'une nouvelle colonne dans df avec la valeur precedent.

edge_roberts = roberts(Img) #appliquer la detection de contrours roberts
edge_roberts1 = edge_roberts.reshape(-1) # convert 1D
df['Roberts'] = edge_roberts1 #Ajout d'une nouvelle colonne dans df avec la valeur precedent.

edge_sobel = sobel(Img)
edge_sobel1 = edge_sobel.reshape(-1) # convert 1D
df['Sobel'] = edge_sobel1 #Ajout d'une nouvelle colonne dans df avec la valeur precedent.

edge_scharr = scharr(Img)
edge_scharr1 = edge_scharr.reshape(-1) # convert 1D
df['Scharr'] = edge_scharr1 #Ajout d'une nouvelle colonne dans df avec la valeur precedent.

edge_prewitt = prewitt(Img)
edge_prewitt1 = edge_prewitt.reshape(-1) # convert 1D
df['Prewitt'] = edge_prewitt1 #Ajout d'une nouvelle colonne dans df avec la valeur precedent.

gaussian_img = nd.gaussian_filter(Img, sigma=3)
gaussian_img1 = gaussian_img.reshape(-1) # convert 1D
df['Gaussian s3'] = gaussian_img1

gaussian_img2 = nd.gaussian_filter(Img, sigma=7) # filtre gaussien
gaussian_img3 = gaussian_img2.reshape(-1) # convert 1D
df['Gaussian s7'] = gaussian_img3

median_img = nd.median_filter(Img, size=3) # filtre median
median_img1 = median_img.reshape(-1) # convert 1D
df['Median s3'] = median_img1
```

### Ajout des labels

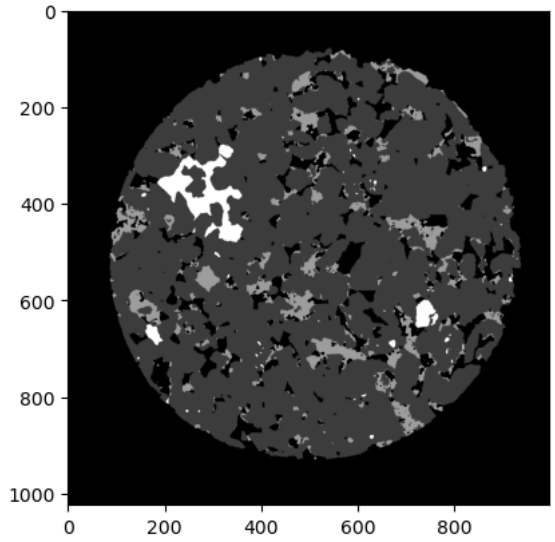
Dans cette partie nous allons ajouter une label à chaque attributs. Ces labels correspondent aux valeurs des pixels de l'image train_masks

Entrée [6]:

```
labeled_img = cv2.imread("C:/Users/Latitude 7280/Desktop/TP ML/TP4/Train_masks/Sandstone_Versa0000.tif") # masque
labeled_img = cv2.cvtColor(labeled_img, cv2.COLOR_BGR2GRAY) # convert gris
labeled_img1 = labeled_img.reshape(-1) # 1D
df['Labels'] = labeled_img1 # ajout dans dataframe df
plt.imshow(labeled_img,'gray') # affichage.
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x253bc782910>
```




Entrée [7]:

```
labeled_img1.shape
```

Out[7]:

```
(1019904,)
```

Entrée [8]:

```
df.head()
```

Out[8]:

| | Original Imag | Gabor1 | Gabor2 | Gabor3 | Gabor4 | Gabor5 | Gabor6 | Gabor7 | Gabor8 | Gabor9 | ... | Gabor32 | Canny Edge | Roberts | Sobel | Scharr | Prewitt | Gaussian s3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 |

5 rows × 42 columns

Entrée [9]:

```
Y = df['Labels']
X = df.drop('Labels',axis = 1)
```

Entrée [10]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2) # Séparation pour Train et test
```

# C) Entrainement du modèle

Entrée [11]:

```
model = RandomForestClassifier(n_estimators = 100, random_state = 42)
```

Entrée [12]:

```
model.fit(X_train,Y_train)
```

Out[12]:

```
RandomForestClassifier(random_state=42)
```

Entrée [13]:

```
model.score(X_test,Y_test)
```

Out[13]:

```
0.9836651452831391
```

Entrée [14]:

```
accuracy_score(Y_train,model.predict(X_train))
```

Out[14]:

```
0.9999987743941524
```

Entrée [15]:

```
accuracy_score(Y_test,model.predict(X_test))
```

Out[15]:

```
0.9836651452831391
```

Une façon d'évaluer la performance d'un modèle de classification est de calculer son score de précision, c'est-à-dire le rapport entre le nombre de prédictions correctes et le nombre total de prédictions. Pour cela, on peut utiliser la méthode model.score ou la fonction accuracy_score du module sklearn.metrics. Ces deux approches donnent le même résultat sur les données du test, comme on peut le voir ci-dessous.

Entrée [16]:

```
features_list = list(X.columns)
fearure_imp = pd.Series(model.feature_importances_,index = features_list).sort_values(ascending=False)
print(fearure_imp)
```

```
Gabor4           1.459814e-01
Gaussian s3      1.054738e-01
Median s3        9.590181e-02
Gabor8           9.458415e-02
Gabor6           8.068042e-02
Gabor5           6.307294e-02
Gabor7           6.284125e-02
Gabor12          5.629746e-02
Gabor24          4.947813e-02
Original Imag    4.309056e-02
Gaussian s7      3.534386e-02
Gabor23          3.337773e-02
Gabor21          1.790990e-02
Gabor30          1.707271e-02
Gabor22          1.705529e-02
Gabor11          1.679497e-02
Gabor29          1.057576e-02
Gabor31          1.001097e-02
Prewitt          9.442710e-03
Scharr           8.463027e-03
Gabor3           8.144037e-03
Sobel            8.128091e-03
Roberts          5.701523e-03
Gabor32          3.477344e-03
Canny Edge       9.747094e-04
Gabor20          1.182276e-04
Gabor28          6.461274e-06
Gabor27          7.696209e-07
Gabor19          3.377683e-08
Gabor2           0.000000e+00
Gabor10          0.000000e+00
Gabor26          0.000000e+00
Gabor1           0.000000e+00
Gabor18          0.000000e+00
Gabor17          0.000000e+00
Gabor16          0.000000e+00
Gabor15          0.000000e+00
Gabor14          0.000000e+00
Gabor13          0.000000e+00
Gabor9           0.000000e+00
Gabor25          0.000000e+00
dtype: float64
```

Une analyse des coefficients de features_list montre que les attributs Gabor4, Gaussian s3, Gabor6, Gabor8 et Median s3 ont une valeur plus élevée que les autres, ce qui signifie qu'ils ont une influence plus forte sur la label. Ces attributs capturent des caractéristiques visuelles importantes des images.

## D) Sauvgarde du modèle

Entrée [17]:

```
filename = "sandstone_model"
pickle.dump(model, open(filename, 'wb'))
```

## E) Test du modèle sur de nouvelles données

Entrée [18]:

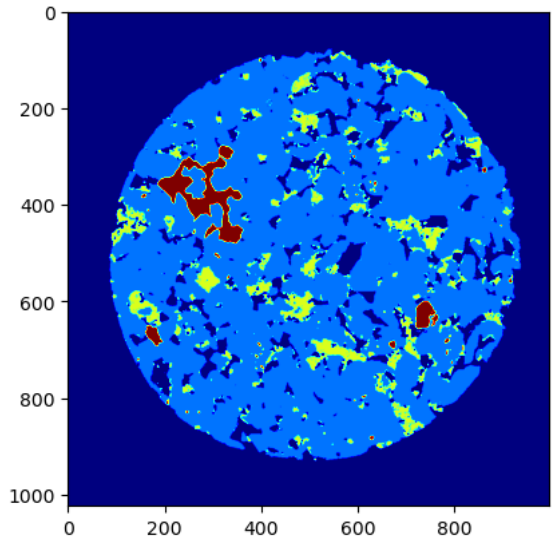
```
loaded_model = pickle.load(open(filename, 'rb')) # load de modele sauvgardé
result = loaded_model.predict(X)
```

Entrée [19]:

```
segmented = result.reshape((Img.shape)) # redimensionnement pour affichage
```

Entrée [20]:

```
plt.imshow(segmented, cmap ='jet') # affiché image segmenté
plt.imsave('segmented_rock_RF_100_estim.jpg', segmented, cmap ='jet') # enregistré l'image.
```




On peut voir que le résultat obtenu est segmenté identiquement à son train masks. Cela signifie que le modèle de segmentation a bien appris à reconnaître les différentes parties d'IRM. On peut également remarquer que les contours des masques sont nets et précis, ce qui indique une bonne qualité de la segmentation. Le modèle a donc réussi à généraliser à partir des données d'entraînement et à produire des masques cohérents.

## Conclusion

Ce travail pratique nous a permis d'explorer les techniques de segmentation d'image en utilisant différents filtres, tels que le filtre de Sobel, le filtre de Canny ou le filtre de prewitt... Nous avons également appris à sauvegarder notre modèle de segmentation pour pouvoir le réutiliser sur d'autres images.