

# Projet d'Apprentissage automatique : Pneumonia\_Detection

***Filière : Ingénierie Numérique en Data Science et Intelligence Artificielle***

***Etablissement : École Nationale supérieure d'Arts et Métiers de Rabat***

***Elément de module : Apprentissage automatique 2***

**Réalisé par :**

**IDELHAJ Amine**

**AARAB ABDELKABIR**

**SOUAD AYOUB**

**Encadré par :**

**Pr. ALQADI Abderrahim**

# Remerciements

Ce travail était effectué au sein de l'*École Nationale supérieure d'Arts et Métiers de Rabat* sous l'encadrement de Professeur ALQADI Abderrahim.

On commence par présenter notre plus vive gratitude à notre encadrant de notre projet de fin d'étude. Mr. ALQADI Abderrahim, Professeur de l'élément du module (Apprentissage automatique). Grâce à ses encouragements, sa pédagogie et ses précieux conseils, il a su nous guider pour mener à bien notre projet. On voudrait exprimer ici notre profonde gratitude à son égard et l'estime respectueuse qu'on lui porte.

On voudrait remercier, toute l'équipe pédagogique et administrative d'École Nationale supérieure d'Arts et Métiers de Rabat, qui nous a aidé par ses qualités humaines, ses rigueurs professionnelles.

On remercie les enseignants chercheurs pour leurs efforts de transmettre aux étudiants leurs connaissances professionnelles.

# TABLE DES MATIERES

## Contents

<b>Remerciements .....</b>	<b>2</b>
<b>TABLE DES MATIERES.....</b>	<b>3</b>
<b>Introduction Générale .....</b>	<b>4</b>
<b>Partie 1 : Modèle d'apprentissage automatique.....</b>	<b>5</b>
I. <b>Analyse des données.....</b>	<b>5</b>
II. <b>Chargement du jeu de données.....</b>	<b>5</b>
III. <b>Visualisation et prétraitement des données.....</b>	<b>5</b>
IV. <b>Aperçu des images des deux classes.....</b>	<b>6</b>
V. <b>Augmentation des données.....</b>	<b>6</b>
VI. <b>Training the Model.....</b>	<b>7</b>
VII. <b>Analyse après la formation du modèle.....</b>	<b>9</b>
VIII. <b>Matrice de confusion.....</b>	<b>10</b>
<b>Partie 2 : Implémentation–Application Django</b> <b>(Pneumonia_Detection) .....</b>	<b>11</b>

## Introduction Générale

L'époque où les données sur les soins de santé étaient limitées est révolue depuis longtemps. Le niveau d'avancement des dispositifs d'acquisition d'images est assez important et c'est ce qui rend le traitement des images difficile et fascinant. Cette croissance significative des images et des techniques médicales nécessite des efforts complets et exhaustifs de la part d'un professionnel de la santé susceptible d'erreur humaine et le résultat peut également varier considérablement d'un expert à l'autre. L'alternative à cette approche consiste à utiliser des stratégies d'apprentissage automatique ou d'apprentissage en profondeur pour automatiser le processus de détection de diverses maladies.

L'apprentissage automatique (ML) et l'intelligence artificielle (IA) ont fait des progrès significatifs au cours des dernières années. Les techniques d'apprentissage automatique et d'intelligence artificielle ont influencé des domaines médicaux tels que le traitement d'images médicales, la reconnaissance d'images, le diagnostic assisté par ordinateur, la segmentation d'images et la fusion d'images, pour n'en nommer que quelques-uns. Alors que la détection automatisée des maladies basée sur les méthodes d'imagerie médicale conventionnelles a démontré des précisions significatives pendant des décennies, les percées dans les approches d'apprentissage automatique ont déclenché une croissance de l'apprentissage en profondeur.

Des algorithmes basés sur l'apprentissage en profondeur ont démontré des résultats remarquables dans divers domaines tels que le diagnostic assisté par ordinateur, la reconnaissance vocale, etc. et garantit également qu'il est exempt d'erreur. Ce faisant, nous voulions minimiser les efforts humains nécessaires pour détecter un rapport de test médical. Nous avons essayé de rendre le système convivial à l'aide de l'interface graphique, afin qu'il puisse être utilisé non seulement par les professionnels de la santé, mais aussi par la population en général.

## **Partie 1 : Modèle d'apprentissage automatique**

## I. Analyse des données

Le jeu de données est organisé en 3 dossiers (train, test, val) et contient des sous-dossiers pour chaque catégorie d'image (Pneumonia/Normal). Il y a 5 863 images radiographiques (JPEG) et 2 catégories (Pneumonie/Normal). Des images radiographiques thoraciques (antéro-postérieures) ont été sélectionnées à partir de cohortes rétrospectives de patients pédiatriques âgés de un à cinq ans du Guangzhou Women and Children's Medical Center, Guangzhou. Toutes les radiographies pulmonaires ont été réalisées dans le cadre des soins cliniques de routine des patients. Pour l'analyse des images radiographiques pulmonaires, toutes les radiographies pulmonaires ont été initialement examinées pour le contrôle de la qualité en supprimant tous les scans de mauvaise qualité ou illisibles. Les diagnostics des images ont ensuite été évalués par deux médecins experts avant d'être autorisés à former le système d'IA. Afin de tenir compte d'éventuelles erreurs de notation, l'ensemble d'évaluation a également été vérifié par un troisième expert.

## II. Chargement du jeu de données

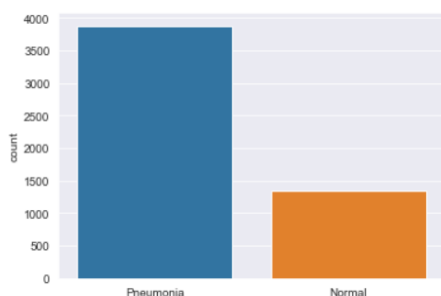
```
train = get_training_data('D:/ChestXRay2017/chest_xray/train')
test = get_training_data('D:/ChestXRay2017/chest_xray/test')
val = get_training_data("D:\\ChestXRay2017\\chest_xray\\val")
```

## III. Visualisation et prétraitement des données

```
Entrée [22]: l = []
for i in train:
    if(i[1] == 0):
        l.append("Pneumonia")
    else:
        l.append("Normal")
sns.set_style('darkgrid')
sns.countplot(l)
```

C:\Users\hp\anaconda3\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 misinterpretation.

Out[22]: <AxesSubplot:ylabel='count'>

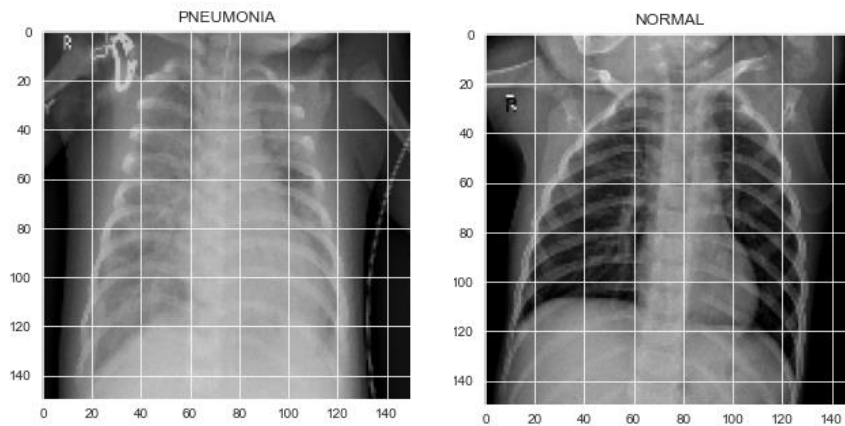


Les données semblent déséquilibrées. Pour augmenter on a utilisé l'augmentation de données.

## IV. Aperçu des images des deux classes

```
plt.figure(figsize = (5,5))
plt.imshow(train[0][0], cmap='gray')
plt.title(labels[train[0][1]])

plt.figure(figsize = (5,5))
plt.imshow(train[-1][0], cmap='gray')
plt.title(labels[train[-1][1]])
```



Nous effectuons une normalisation des niveaux de gris pour réduire l'effet des différences d'illumination. De plus, le CNN converge plus rapidement sur les données [0..1] que sur [0..255].

```
# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255

# resize data for deep learning
x_train = x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val = x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)

x_test = x_test.reshape(-1, img_size, img_size, 1)
y_test = np.array(y_test)
```

## V. Augmentation des données

Afin d'éviter les problèmes de sur ajustement, nous devons étendre artificiellement notre ensemble de données. Nous pouvons rendre votre ensemble de données existant encore plus grand. L'idée est de modifier les données d'entraînement avec de petites transformations pour reproduire les variations. Les approches qui modifient les données d'apprentissage de manière à modifier la représentation du tableau tout en conservant la même étiquette sont appelées techniques d'augmentation des données. Certaines augmentations populaires que les gens utilisent sont les niveaux de gris, les retournements horizontaux, les retournements verticaux, les cultures aléatoires, les variations de couleur, les traductions, les rotations et bien plus encore. En appliquant seulement quelques-unes de ces transformations à nos données d'entraînement, nous pouvons facilement doubler ou

tripler le nombre d'exemples d'entraînement et créer un modèle très robuste.

```
# With data augmentation to prevent overfitting and handling the imbalance in dataset

datagen = ImageDataGenerator(
    featurewise_center=False, # set input mean to 0 over the dataset
    samplewise_center=False, # set each sample mean to 0
    featurewise_std_normalization=False, # divide inputs by std of the dataset
    samplewise_std_normalization=False, # divide each input by its std
    zca_whitening=False, # apply ZCA whitening
    rotation_range = 30, # randomly rotate images in the range (degrees, 0 to 180)
    zoom_range = 0.2, # Randomly zoom image
    width_shift_range=0.1, # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1, # randomly shift images vertically (fraction of total height)
    horizontal_flip = True, # randomly flip images
    vertical_flip=False) # randomly flip images

datagen.fit(x_train)
```

Pour l'augmentation des données, on a choisi de :

1. Faire pivoter au hasard certaines images d'entraînement de 30 degrés.
2. Zoom aléatoire de 20 % sur certaines images d'entraînement.
3. Décaler aléatoirement les images horizontalement de 10 % de la largeur.
4. Décaler aléatoirement les images verticalement de 10 % de la hauteur.
5. Retourner aléatoirement les images horizontalement.

Une fois que notre modèle est prêt, nous adaptons l'ensemble de données d'entraînement.

## VI. Training the Model

```
model = Sequential()
model.add(Conv2D(32, (3,3), strides = 1, padding = 'same', activation = 'relu', input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides = 2, padding = 'same'))
model.add(Conv2D(64, (3,3), strides = 1, padding = 'same', activation = 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides = 2, padding = 'same'))
model.add(Conv2D(64, (3,3), strides = 1, padding = 'same', activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides = 2, padding = 'same'))
model.add(Conv2D(128, (3,3), strides = 1, padding = 'same', activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides = 2, padding = 'same'))
model.add(Conv2D(256, (3,3), strides = 1, padding = 'same', activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2), strides = 2, padding = 'same'))
model.add(Flatten())
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1, activation = 'sigmoid'))
model.compile(optimizer = "rmsprop", loss = 'binary_crossentropy', metrics = ['accuracy'])
model.summary()
```

### IV.1 Model Predicting

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience = 2, verbose=1,
factor=0.3, min_lr=0.000001)
```

```
history = model.fit(datagen.flow(x_train,y_train, batch_size = 32) ,epochs = 12 , validation
_data = datagen.flow(x_val, y_val) ,callbacks = [learning_rate_reduction])
```

## IV.2 Resultat de prédiction

```
Epoch 1/12
164/164 [=====] - 214s 1s/step - loss: 0.2864 - accuracy: 0.8930 - val_loss: 38.0237 - val_accuracy:
0.5000 - lr: 0.0010
Epoch 2/12
164/164 [=====] - 224s 1s/step - loss: 0.2442 - accuracy: 0.9117 - val_loss: 5.7720 - val_accuracy: 0.
5000 - lr: 0.0010
Epoch 3/12
164/164 [=====] - ETA: 0s - loss: 0.2150 - accuracy: 0.9203
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
164/164 [=====] - 244s 1s/step - loss: 0.2150 - accuracy: 0.9203 - val_loss: 36.0529 - val_accuracy:
0.5000 - lr: 0.0010
Epoch 4/12
164/164 [=====] - 216s 1s/step - loss: 0.1488 - accuracy: 0.9450 - val_loss: 0.3327 - val_accuracy: 0.
8750 - lr: 3.0000e-04
Epoch 5/12
164/164 [=====] - 189s 1s/step - loss: 0.1421 - accuracy: 0.9509 - val_loss: 0.9149 - val_accuracy: 0.
5625 - lr: 3.0000e-04
Epoch 6/12
164/164 [=====] - ETA: 0s - loss: 0.1311 - accuracy: 0.9522
Epoch 6: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
164/164 [=====] - 192s 1s/step - loss: 0.1311 - accuracy: 0.9522 - val_loss: 7.0899 - val_accuracy: 0.
5000 - lr: 3.0000e-04
Epoch 7/12
164/164 [=====] - 188s 1s/step - loss: 0.1215 - accuracy: 0.9597 - val_loss: 1.0093 - val_accuracy: 0.
5625 - lr: 9.0000e-05
Epoch 8/12
164/164 [=====] - ETA: 0s - loss: 0.1043 - accuracy: 0.9639
Epoch 8: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
164/164 [=====] - 187s 1s/step - loss: 0.1043 - accuracy: 0.9639 - val_loss: 2.3002 - val_accuracy: 0.
5625 - lr: 9.0000e-05
Epoch 9/12
164/164 [=====] - 187s 1s/step - loss: 0.1025 - accuracy: 0.9622 - val_loss: 0.4637 - val_accuracy: 0.
8125 - lr: 2.7000e-05
Epoch 10/12
164/164 [=====] - ETA: 0s - loss: 0.1040 - accuracy: 0.9631
Epoch 10: ReduceLROnPlateau reducing learning rate to 8.10000001365517e-06.
164/164 [=====] - 188s 1s/step - loss: 0.1040 - accuracy: 0.9631 - val_loss: 0.4375 - val_accuracy: 0.
7500 - lr: 2.7000e-05
Epoch 11/12
164/164 [=====] - 217s 1s/step - loss: 0.0947 - accuracy: 0.9648 - val_loss: 1.4040 - val_accuracy: 0.
6250 - lr: 8.1000e-06
Epoch 12/12
164/164 [=====] - ETA: 0s - loss: 0.1058 - accuracy: 0.9681
Epoch 12: ReduceLROnPlateau reducing learning rate to 2.429999949526973e-06.
164/164 [=====] - 206s 1s/step - loss: 0.1058 - accuracy: 0.9681 - val_loss: 1.1623 - val_accuracy: 0.
5625 - lr: 8.1000e-06
```

```
print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 , "%")
```

```
20/20 [=====] - 4s 183ms/step - loss: 0.2534 - accuracy: 0.9103
Loss of the model is - 0.25344812870025635
20/20 [=====] - 4s 183ms/step - loss: 0.2534 - accuracy: 0.9103
Accuracy of the model is - 91.02563858032227 %
```

On constat que notre prédiction est presque 92% donc notre modèle est validé.

## VII. Analyse après la formation du modèle.



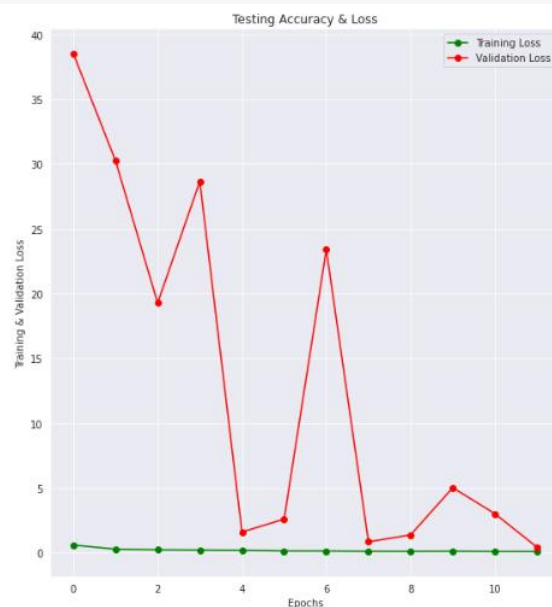
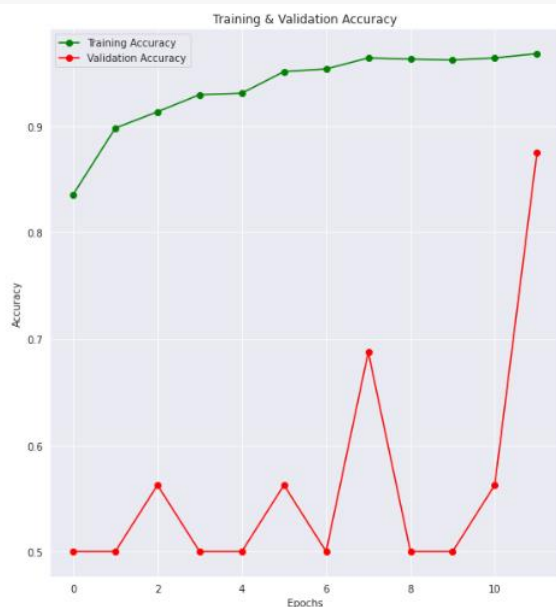
```

epochs = [i for i in range(12)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20,10)

ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')
ax[0].plot(epochs , val_acc , 'ro-' , label = 'Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs , train_loss , 'g-o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-o' , label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()

```



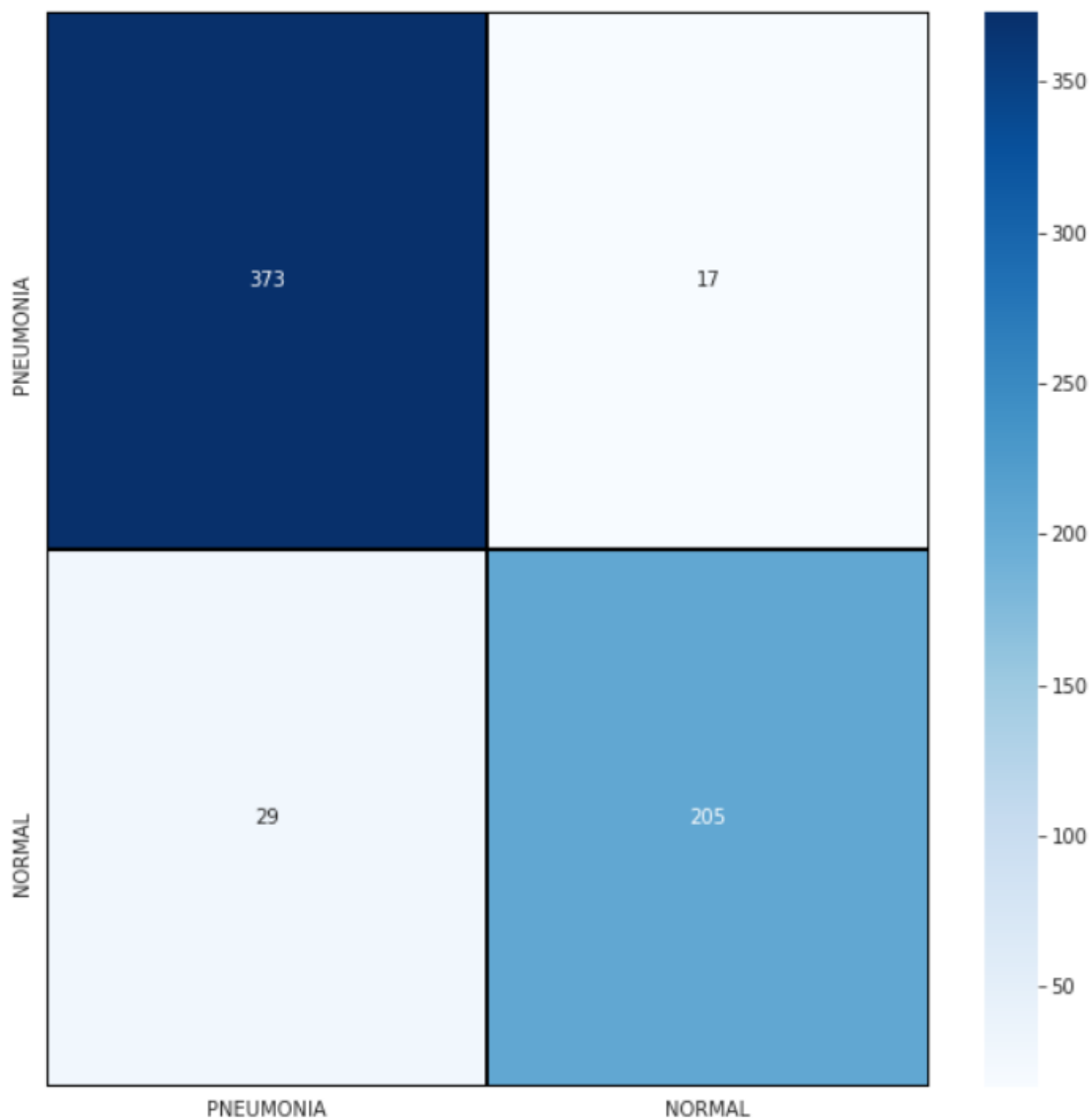
## VIII. Matrice de confusion.

```
cm = confusion_matrix(y_test,predictions)
cm
```

```
array([[373, 17],
       [ 29, 205]])
```

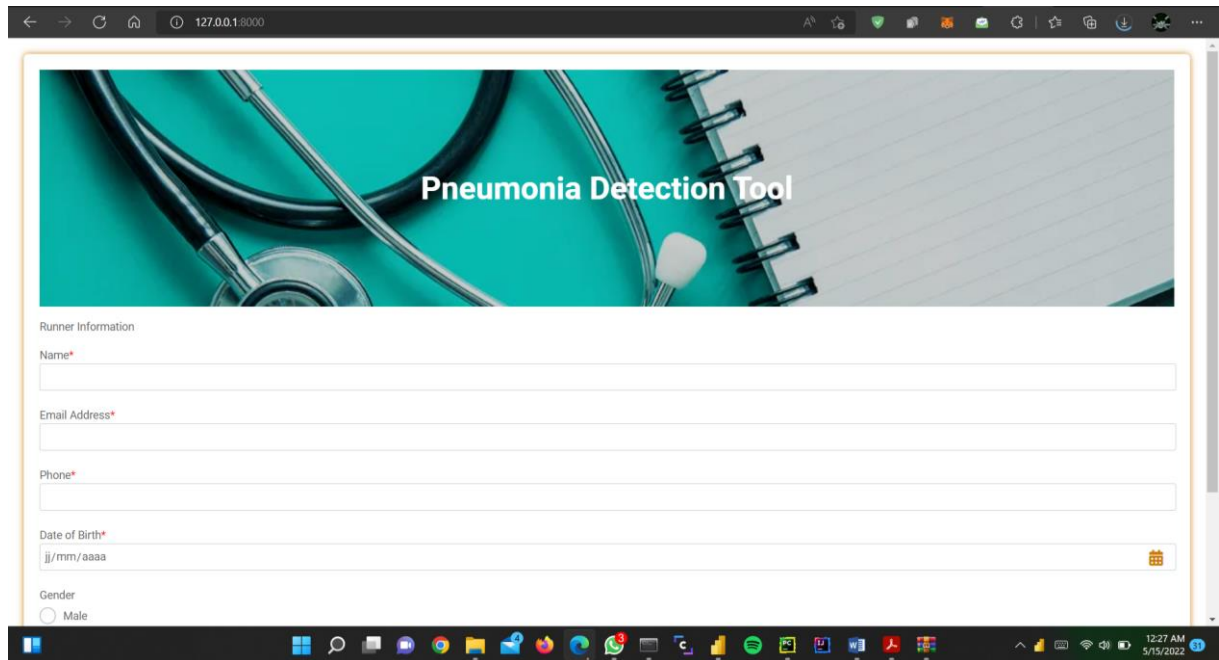
```
cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1'])
```

```
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot = True, fmt='',xti
cklabels = labels,yticklabels = labels)
```



## Partie 2 : Implémentation–Application Django

## (Pneumonia\_Detection)



The screenshot displays a web browser window with the address bar showing '127.0.0.1:6000'. The main content area has a header image with a stethoscope and a notebook, overlaid with the text 'Pneumonia Detection Tool'. Below the header is a form titled 'Runner Information' with the following fields:

- Name\*
- Email Address\*
- Phone\*
- Date of Birth\* (format: jj/mm/aaaa)
- Gender (radio button for Male)

The Windows taskbar is visible at the bottom of the screen, showing various application icons and the system clock indicating 12:27 AM on 5/15/2022.

L'interface sous forme d'un formulaire que l'utilisateur doit remplir puis il doit insérer l'image pour que le modèle prédise si l'utilisateur est malade par la pneumonie ou non.