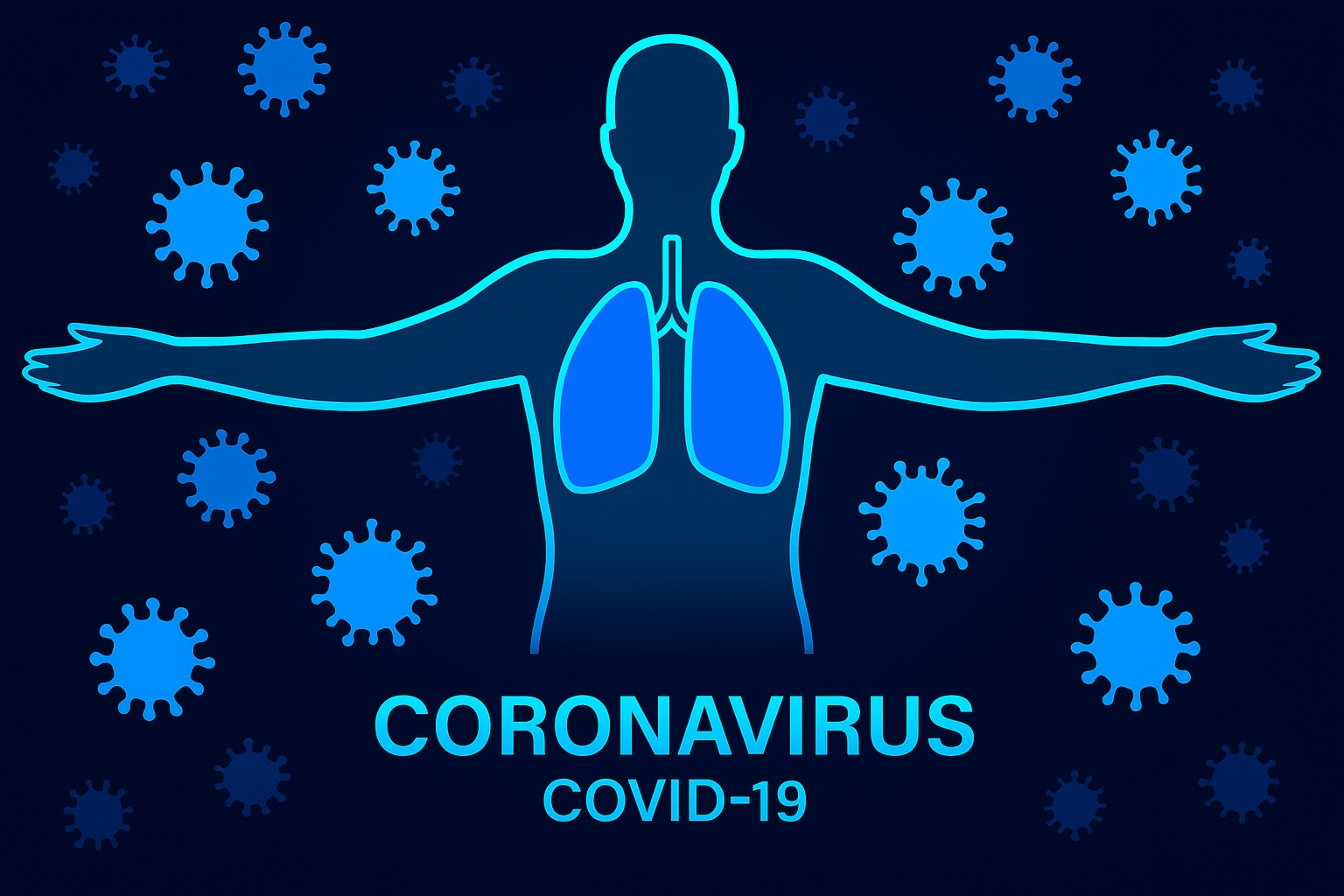


**Analyse de Radiographies pulmonaires Covid-19**



**Présenté par : Encadré par :**

* Cirine Bouamrane Nicolas Mormiche
* Léna Bacot
* Steven Moire
* Rafael Cepa

**Sommaire :**

[Introduction : 2](#_Toc1179674613)

[Etape 1 : Exploration des données et DataViz’ 3](#_Toc2021026848)

[1.1 Base de données : 3](#_Toc2094333101)

[1.2 Dataviz’ : 4](#_Toc178994952)

[1.2.1 Histogramme de la luminosité et du contraste : 4](#_Toc622789717)

[Interprétation : 6](#_Toc1867174697)

[Luminosité Globale : 7](#_Toc1903797319)

[Contraste Globale : 7](#_Toc1288651542)

[1.2.2 Visualisation des images en L et en RGB : 7](#_Toc588370307)

[Recommandations techniques : 9](#_Toc739841953)

[Template : 9](#_Toc113043947)

[1.2.3 Stream lit Graph 10](#_Toc423555603)

[Introduction : 10](#_Toc1648698605)

[Outils Collaboratifs : 10](#_Toc231260904)

[Inspection des Fichiers Features : 10](#_Toc2022066903)

[Inspection détaillée des fonctions dans Features : 10](#_Toc1877804247)

[Analyse Complète (Déséquilibrée) : 11](#_Toc2110262772)

[Analyse Equilibrée : 15](#_Toc2065494281)

[1.2.4 Génération images masquées + Statistiques : 18](#_Toc1733749331)

[Exemple Génération Image Masqué : 18](#_Toc2080553922)

[Etape 02: Pre-processing et features engineering 19](#_Toc741846769)

[2.1 Redimensionnement et conversion en niveaux de gris : 19](#_Toc1875429090)

[2.2 Normalisation des pixels : 20](#_Toc752344774)

[2.3 Augmentation data 21](#_Toc129429574)

[2.3.1 Sous échantillonnage : (code en annexe 02) 21](#_Toc452517779)

[2.3.2 Sur échantillonnage : (code en annexe 04) 21](#_Toc323226062)

[3.1 Model Baseline 22](#_Toc925762981)

[3.1.1 Modele Machine learning: 22](#_Toc39311318)

[3.2 Deep Learning : 26](#_Toc949934968)

[Interprétation : 28](#_Toc962920922)

[Nous avons aussi calculé la matrice de confusion,les résultats de la matrice de confusion obtenus pour le modèle de deep learning sont représentés dans la figure 38. 29](#_Toc1679306826)

[Figure 38 : les résultats de la matrice de confusion pour le modèle de deep learning 29](#_Toc170860727)

[La matrice de confusion générée par le modèle de deep learning montrent que notre modèle est très performant et qu’il arrive a bien identifié les différentes classes : COVID, pneumonia, virale et état normal. 29](#_Toc1724458396)

[3.3 Optimisation : 29](#_Toc712486815)

[3.3.1 Grid search: 29](#_Toc1785174717)

[Annexe : 33](#_Toc1085183884)

[Annexe 1 : Sélecteur de dossiers pour Jupyter 34](#_Toc774354770)

[Annexe 2 : Undersampling d’images 34](#_Toc414917403)

[Annexe 3 : Pipeline principal 35](#_Toc587415588)

[Annexe 4 : Oversampling avec Keras 36](#_Toc1775489080)

# **Introduction :**

La pandémie de COVID-19, causée par le virus SARS-CoV-2, a constitué un défi sanitaire mondial majeur avec plus de 243 millions de cas et 4,9 millions de décès à ce jour. Face à cette urgence, l'identification rapide et précise des cas positifs a été essentielle pour maîtriser la propagation et optimiser la prise en charge des patients. Les tests RT-PCR, bien que standards, ont présenté des limites comme un taux élevé de faux négatifs et des délais pour obtenir les résultats. En complément, l’imagerie médicale, notamment la tomodensitométrie (CT), a été largement adoptée, mais elle reste coûteuse et expose à une irradiation notable.

Ainsi, la radiographie thoracique (CXR) est apparue comme une alternative plus accessible, rapide et moins irradiant les patients, bien que son interprétation soit complexe en raison de similitudes radiologiques avec d'autres pneumonies virales et pathologies pulmonaires, ce qui peut entraîner des erreurs diagnostiques. L’intelligence artificielle et l’apprentissage profond, en particulier via les réseaux de neurones convolutionnels, ont permis des avancées significatives dans l’analyse automatisée des radiographies thoraciques COVID-19, montrant des performances prometteuses malgré des limitations liées à la taille des datasets et au sur-apprentissage.

Notre projet s’inscrira dans cette dynamique en visant à développer une méthode robuste intégrant dès la première étape une étude approfondie des données dans la partie datavisualisation, permettant de comprendre leur distribution et caractéristiques. Ensuite, nous procèderons au prétraitement comprenant redimensionnement, conversion en niveaux de gris, normalisation des pixels et augmentation des données pour enrichir le dataset et limiter le surapprentissage.

En modélisation, nous testerons trois modèles classiques de machine learning : SVM, KNN et Random Forest. Puis, en deep learning, nous utiliserons un modèle baseline, InceptionV3 pré-entraîné sur ImageNet, adapté à notre tâche. Durant la phase d’optimisation, nous améliorerons les performances des modèles de machine learning grâce à la recherche sur grille (grid search) et en expérimentant avec le classificateur XGBoost.

Pour finir, nous appliquerons plusieurs méthodes d’interprétabilité sur les résultats obtenus avec le modèle deep learning, notamment LIME, SHAP et Grad-CAM, afin de garantir une meilleure compréhension et confiance dans les décisions du système.

Cette démarche complète et structurée permettra de combiner rigueur scientifique, performance et interprétabilité pour une application fiable en contexte médical, facilitant son adoption même dans des environnements avec des ressources diagnostiques limitées.

# **Etape 1 : Exploration des données et DataViz’**

## **1.1 Base de données :**

La base de données utilisée pour ce projet est la COVID-19 Radiography Database accessible sur Kaggle. Cette base de données regroupe différentes catégories d'images radiographiques pulmonaires permettant la détection et la classification des infections, notamment du Covid-19. Elle inclut :

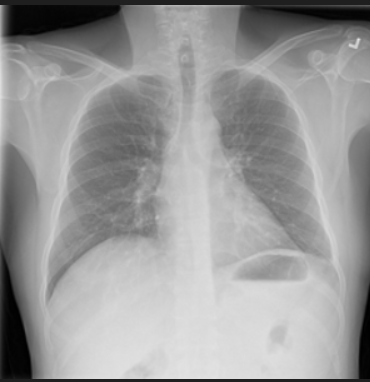
* Des radiographies thoraciques de patients confirmés positifs au Covid-19.
* Des images de cas jugés normaux (sans infection pulmonaire détectée).
* Des radiographies de pneumonies virales autres que Covid-19.
* Des images présentant des opacités pulmonaires (infections pulmonaires non-COVID de type "Lung Opacity").

Les tailles des sous-ensembles composant la base de données sont représentées dans le tableau suivant :

|  |  |  |
| --- | --- | --- |
| **Classes** | **Images** | **Masks** |
| **Covid** | 3616 | 3616 |
| **Normal** | 10192 | 10192 |
| **Viral\_pneumonia** | 1345 | 1345 |
| **Lung\_opacity** | 6012 | 6012 |

Ces images proviennent de multiples sources publiques, publications spécialisées et collaborations avec des hôpitaux.

Un exemple d’une image de la classe normale et son mask sont représentées dans la figure ci-dessous (Figure01). [1]



**Figure 01 :** Exemple d’une image de la base de données et son mask (classe nomal)

## **1.2 Dataviz’ :**

### **1.2.1 Histogramme de la luminosité et du contraste :**

* **Luminosité :**  
  Il s’agit de l’intensité lumineuse globale d’une image, c’est-à-dire du niveau moyen de clarté ou d’obscurité perçu sur l’ensemble de l’image. Mathématiquement, c’est simplement la moyenne des valeurs des canaux de chaque pixel. Pour une image en couleurs (RGB), on utilise la pondération de la norme ITU-R BT.601, correspondant à la sensibilité de l’œil humain (0.299 pour R, 0.587 pour G, 0.114 pour B).[2]
* **Contraste :**  
  Le contraste représente la différence de luminosité entre les zones claires et sombres d’une image. Mathématiquement, c’est l’écart-type (std) de la valeur des pixels. Un contraste élevé signifie que la différence entre les parties les plus sombres et les plus claires est importante ; à l’inverse, un faible contraste indique que les valeurs de gris sont proches les unes des autres, rendant les détails moins visibles.[3]

Un histogramme de luminosité et de contraste fournit des informations essentielles sur la répartition des tons et la qualité d’exposition d’une image, l’histogramme permet de juger de la bonne exposition, du niveau de contraste, de repérer d’éventuelles pertes d’information, et de guider les ajustements à apporter pour optimiser une image.

Il indique comment les pixels de l’image sont répartis des tons les plus foncés (gauche) aux plus clairs (droite) :

* À gauche : pixels sombres (ombres, noirs)
* Au centre : tons moyens (gris)
* À droite : pixels clairs (hautes lumières, blancs)

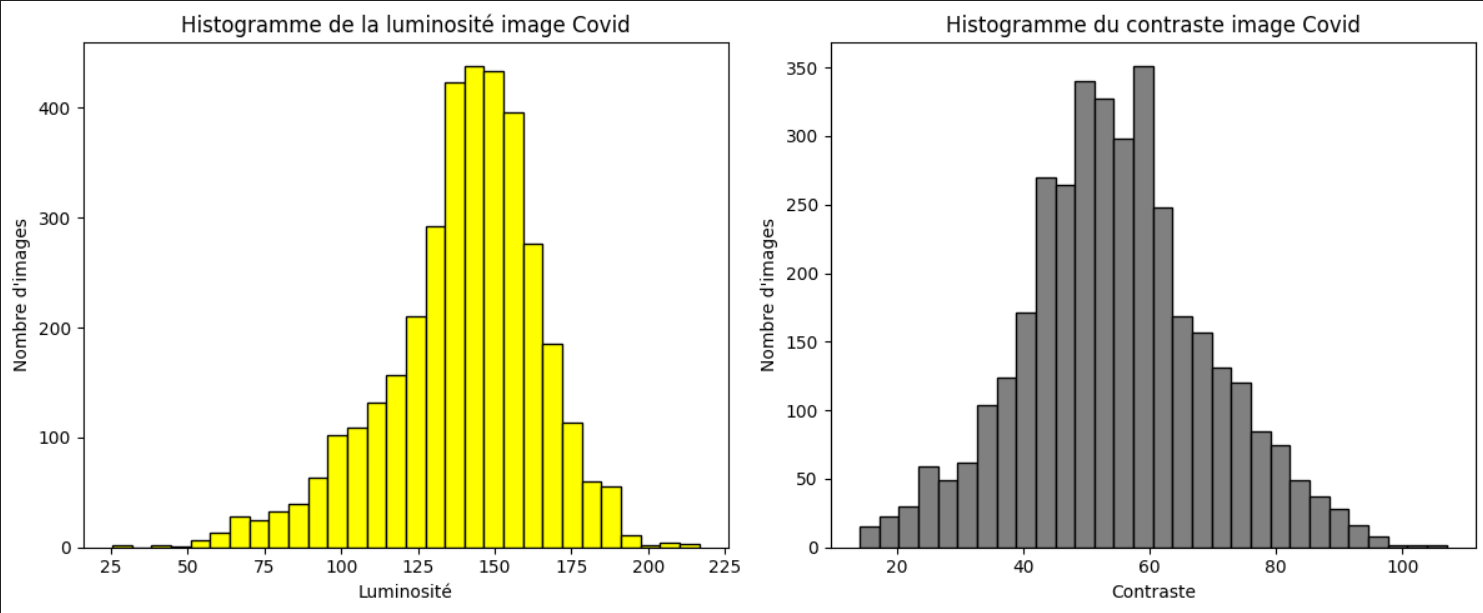
Il permet d’identifier :

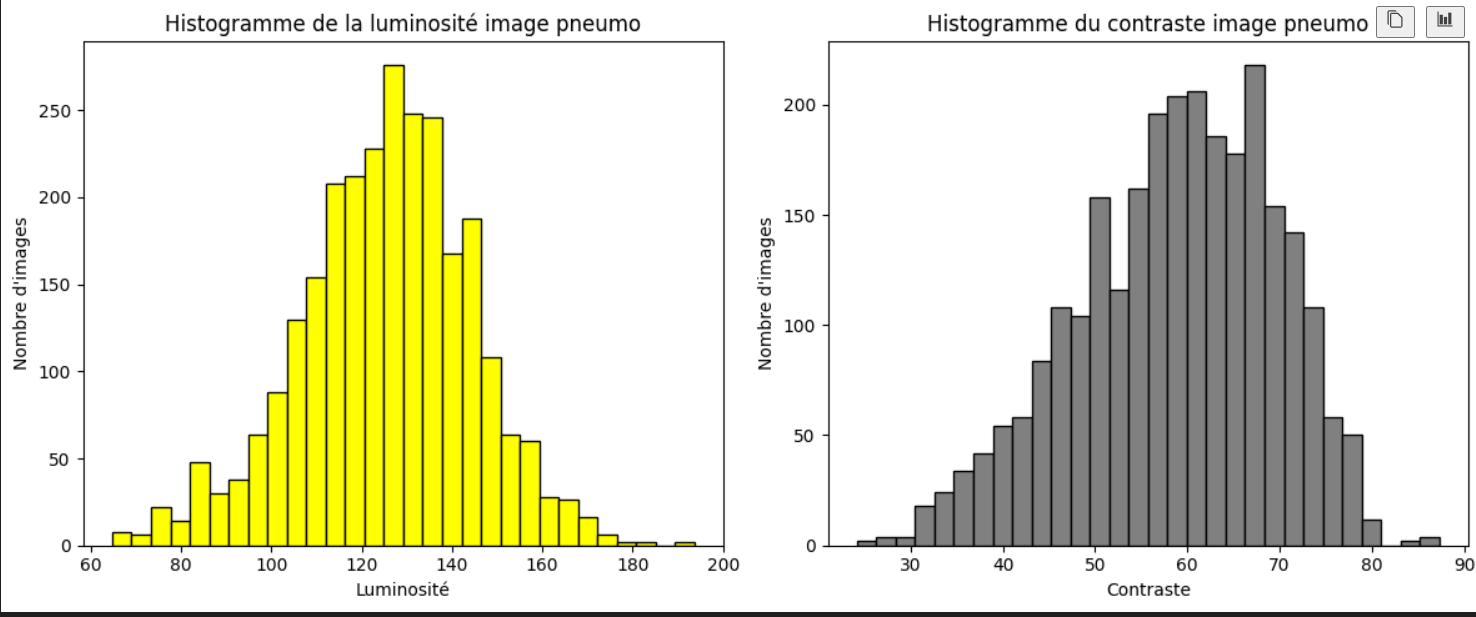
* Une sous-exposition (pic vers la gauche : image trop sombre)
* Une surexposition (pic vers la droite : image trop claire)
* Un bon contraste (répartition étalée de la gauche à la droite : l’image contient à la fois des zones sombres et claires, donc beaucoup de détails)
* Un faible contraste (histogramme ramené vers le centre, l’image paraît “plate” avec surtout des tons moyens)

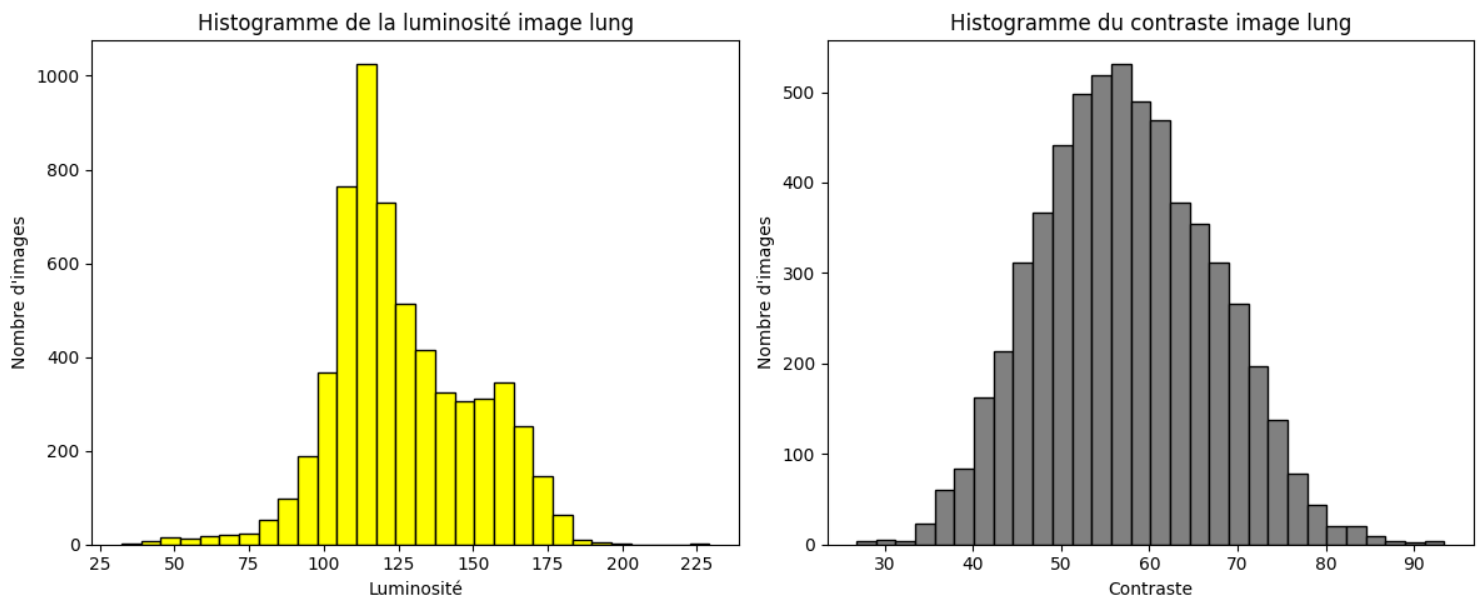
L’histogramme aide aussi à :

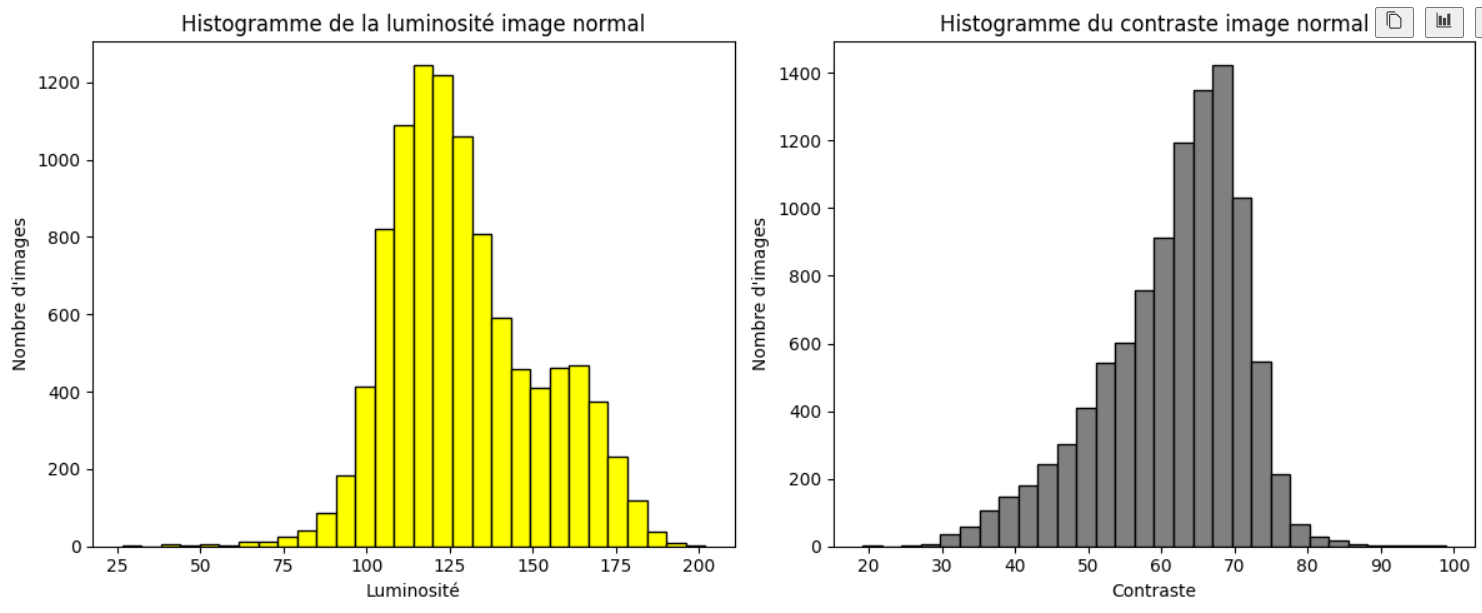
* Éviter l’écrêtage : apparition de barres hautes à l’extrême gauche ou droite, signalant une perte d’information dans les ombres ou les hautes lumières.
* Ajuster la correction de la luminosité et du contraste pour améliorer la lisibilité ou la qualité technique de l’image.[2]

Nous avons généré les histogrammes de luminosité pour chaque catégorie d’images (normal, Covid, pneumonie et lung). Les résultats correspondants sont illustrés dans les figures suivantes :









**Figure 02 :** Histogramme de la luminosité et du contraste des différentes images de la dataset

## **Interprétation :**

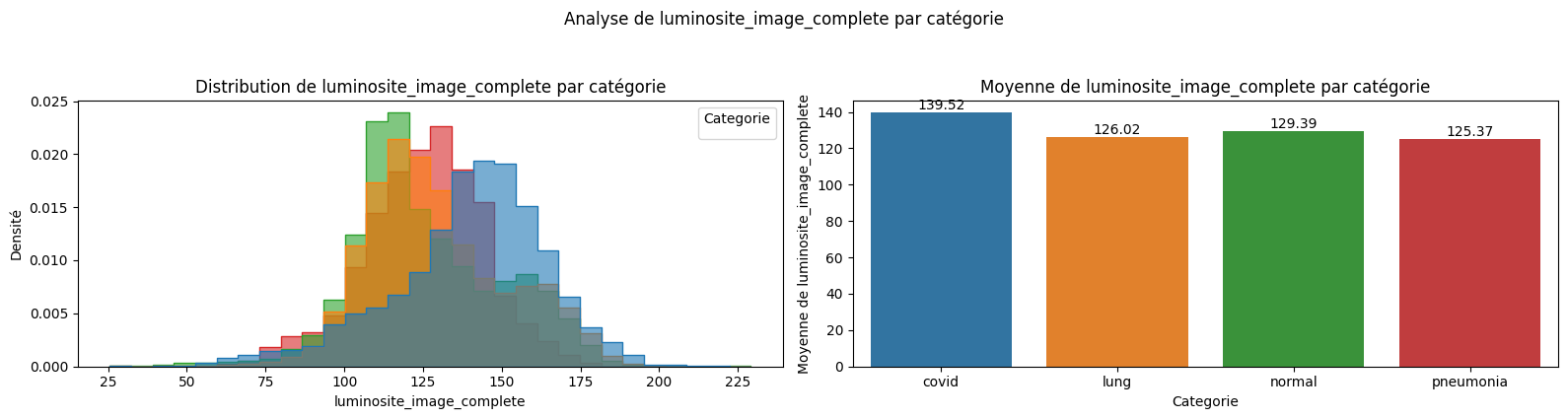
A savoir :

* **Axe horizontal :** lesvaleurs de gris, de 0 (noir) à 255 (blanc).
* **Axe vertical :** nombre d’images.

|  |  |  |
| --- | --- | --- |
|  | **Histogramme luminosité** | **Histogramme contraste** |
| **Image normale** | Ton vers le clair | Faible contraste |
| **Image covid** | Ton vers le clair | Bon contraste |
| **Image Pneumo** | Ton moyen (gris équilibré) | Bon contraste |
| **Image lung** | Ton vers le clair | Bon contraste |

## **Luminosité Globale :**

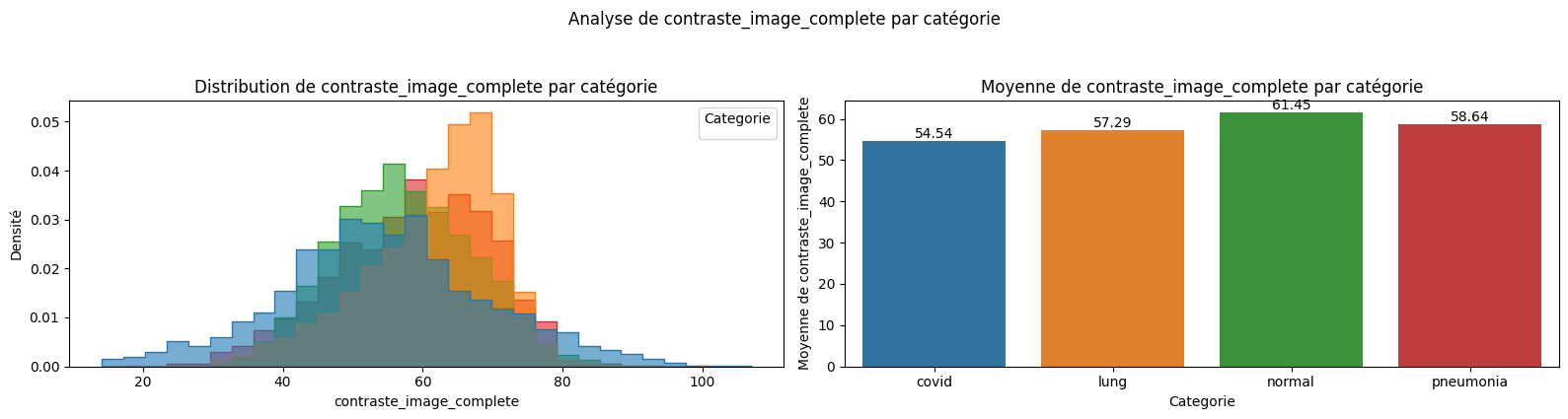
Exprime la répartition de la luminosité des images, pour chaque catégorie.



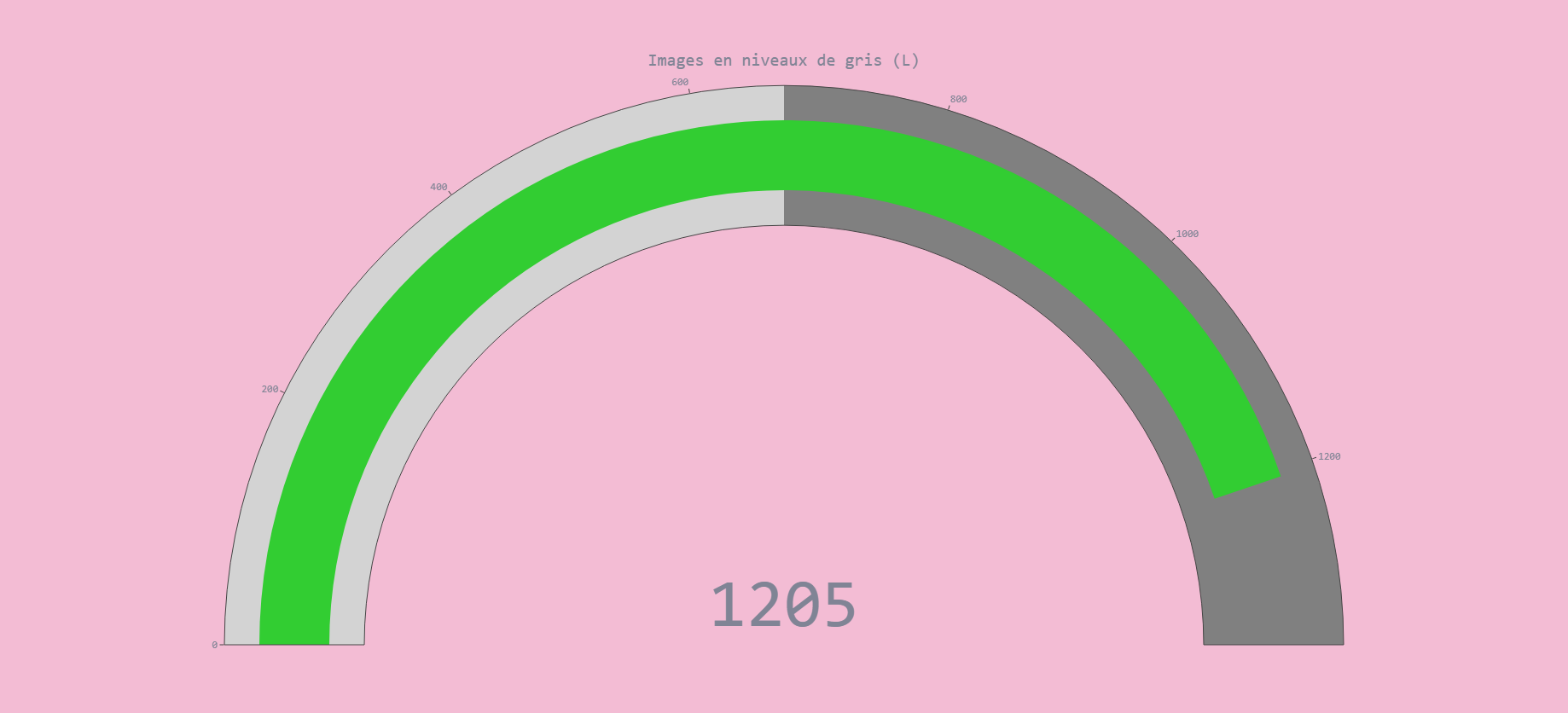
**Figure 03 :** Distribution Luminosité Globale

## **Contraste Globale :**

Exprime la répartition du contraste des images, pour chaque catégorie.

**Figure 04 :** Distribution du contraste Globale

### **1.2.2 Visualisation des images en L et en RGB :**

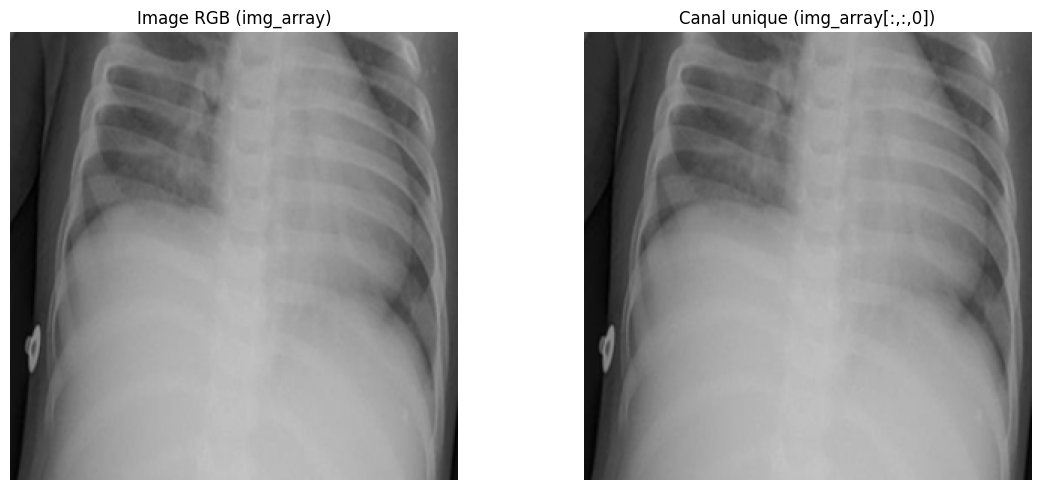


**Figure 05 :** Visualisation des images en L et en RGB de la classe Viral Pneumonia

140 images RGB détectées (hors norme) sur un total de 1345 images provenant de Viral Pneumonia.

Après analyse, nous avons conclu qu’il s’agit en fait de ‘faux RGB’, c’est à dire des images encodées en 3 canaux, mais représentant en réalité une image L. En effet, quand on a essayé d’afficher quelques intrus, nous avions l’impression de voir des niveaux de gris. Pour chaque intrus ‘img’, convertie en np.array, nous avons :

img\_array[:,:,0] == img\_array[:,:,1] == img\_array[:,:,2]



**Figure 06 :** exemple d’images en RGB de la classe Viral Pneumonia

Ces 140 images sont bien codées en 3 canaux RGB, mais on a, à chaque fois, la même valeur pour les trois canaux, c’est donc une redondance d’informations, et la conversion en L sera alors encore plus simple. Pour rappel, une image RGB est convertie en L selon la norme ITU-R BT.601 (citée plus haut), grâce à la formule :

L = 0.299R + 0.587G + 0.114B.

Où :

* L : valeur du pixel en nuance de gris résultante de la conversion
* R, G, B : valeurs du pixel en RGB que l’on veut convertir

Il est donc évident ici, qu’avec R = G = B :

L = 0.299R + 0.587R + 0.114R L = R = G = B

Autrement dit, le pixel converti en L n’est autre que la même valeur commune aux trois channels, on peut donc simplement choisir le premier canal (R, i.e. img\_array[:,:,0]) pour ‘convertir’ l’image en L.

Nous avons remarqué qu’en réalité, ce problème est exactement le même pour les masks, qui sont tous des “faux RGB”. Pour la suite de notre projet, nous avons décidé d’utiliser les fonctions de conversion des packages reconnus (PIL, cv2 ...), même si utiliser le premier canal (par exemple) suffirait. En effet, nous voulons tester différentes approches, et essayer de comprendre ce qu’apportent ces fonctions. Il nous semblait cependant important de montrer notre réflexion face à ce problème.

**Analyse métier**

* Les radiographies doivent être en niveau de gris (L), car les informations pertinentes ne sont pas dans les couleurs.
* La présence de plusieurs images en RGB est donc anormale.
* Cela peut indiquer des erreurs de traitement ou d'export depuis un outil d'annotation.

## **Recommandations techniques :**

* Identifier les images en RGB et les vérifier visuellement.
* Si ce sont bien des radios, les convertir en L via img.convert("L") pour :
  + Homogénéiser les données
  + Réduire la taille mémoire
* Ces anomalies sont visibles dans le rapport template d’exploration automatique généré (voir lien juste après).

## **Template :**

[**https://1drv.ms/x/c/9e322a9006fb4eb1/ERjCTdYjnJBFhoRxyeNj00EBlyoYCRADMprJFlOrgObT0Q?e=idyaqO**](https://1drv.ms/x/c/9e322a9006fb4eb1/ERjCTdYjnJBFhoRxyeNj00EBlyoYCRADMprJFlOrgObT0Q?e=idyaqO)

### **1.2.3 Stream lit Graph**

## **Introduction :**

La bibliothèque StreamLit sur python, nous permet de créer des Dashboard interactifs.

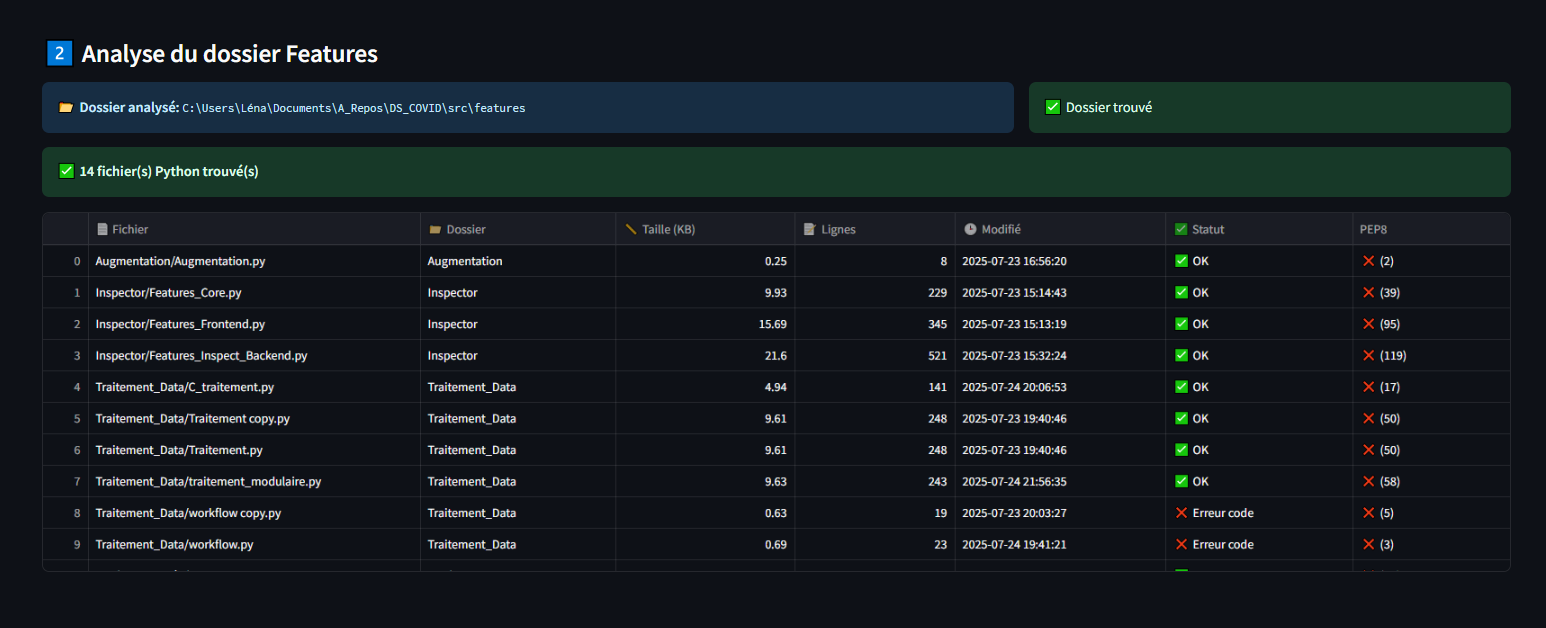
Ceux-ci ont été conçus afin de nous permettre de créer des outils afin de faciliter notre collaboration.

Le Stream Lit peut appeler les fonctions en backend pour effectuer certaines taches dans notre data-workflow et nous présenter des visualisations d’analyses que l’on a effectué.  
  
A la suite, vous trouverez un aperçu de ces fonctionnalités et un cas d’application pratique sur notre problématique.

## **Outils Collaboratifs :**

### **Inspection des Fichiers Features :**

Voici un aperçu de la fenêtre d’inspection des fichiers.



**Figure 07 :** Inspection Fichiers Python Backend

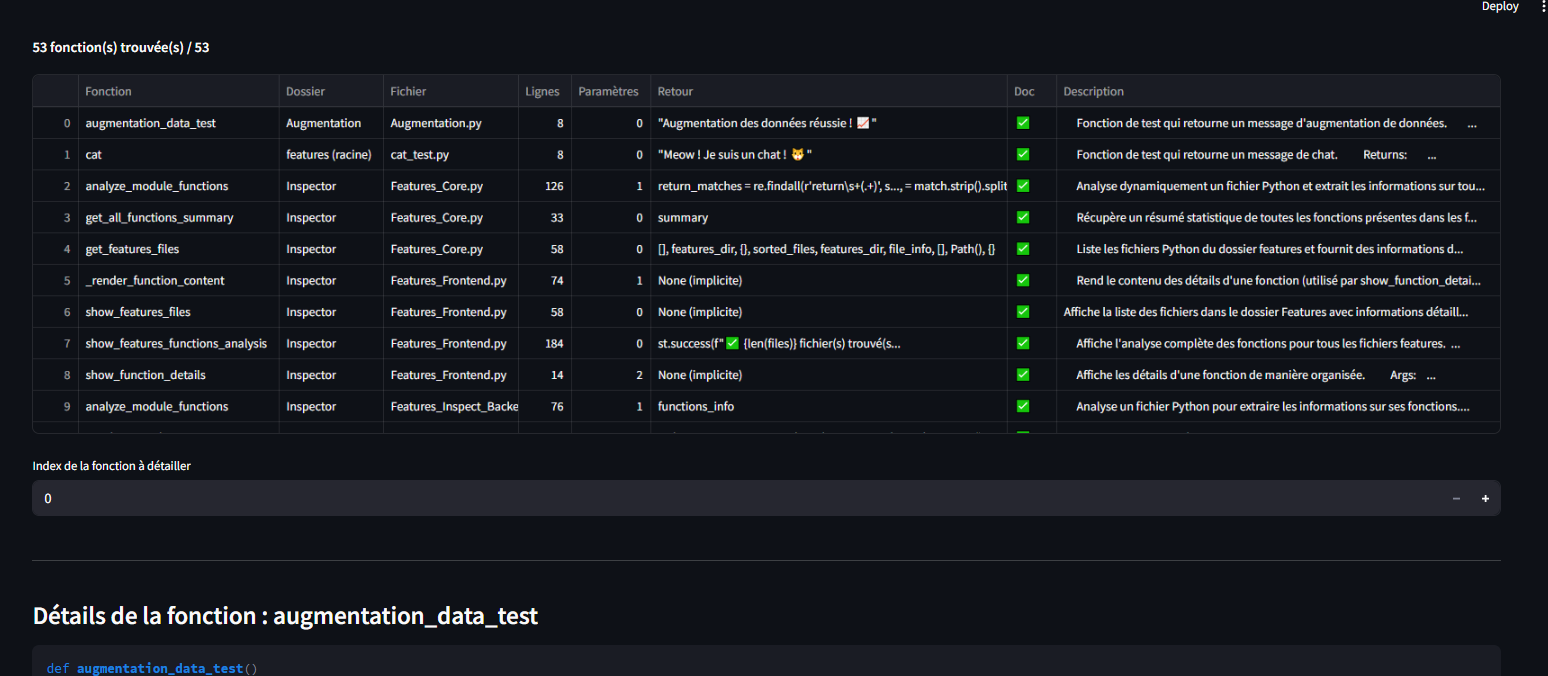
Cet affichage nous permet de voir l’évolution des fichiers de fonctions backend d’un simple coup d’œil !

On peut donc voir le nombre de lignes et la validation PEP8, nous permettant de revoir les fichiers adéquats et donc de gagner en organisation.

### **Inspection détaillée des fonctions dans Features :**

Pour chacun des fichiers détectés précédemment, on vient scanner toutes les fonctions (à l’aide de la librairie Inspect) afin de les afficher.

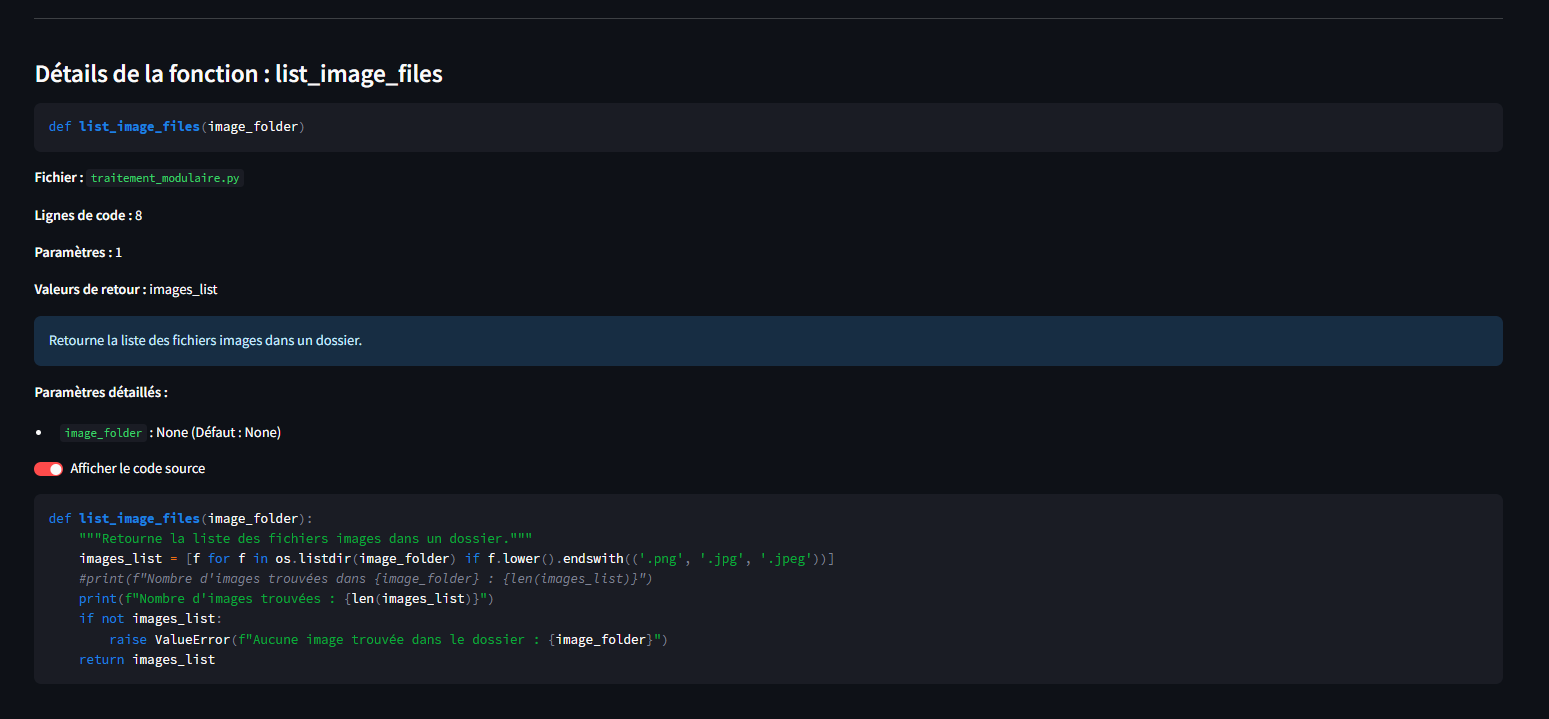
Ceci sert pour le travail collaboratif afin de rester informé de l’existence des fonctions crée ou mises à jour par nos équipes.



**Figure 08:** Inspection Fonctions Python Backend

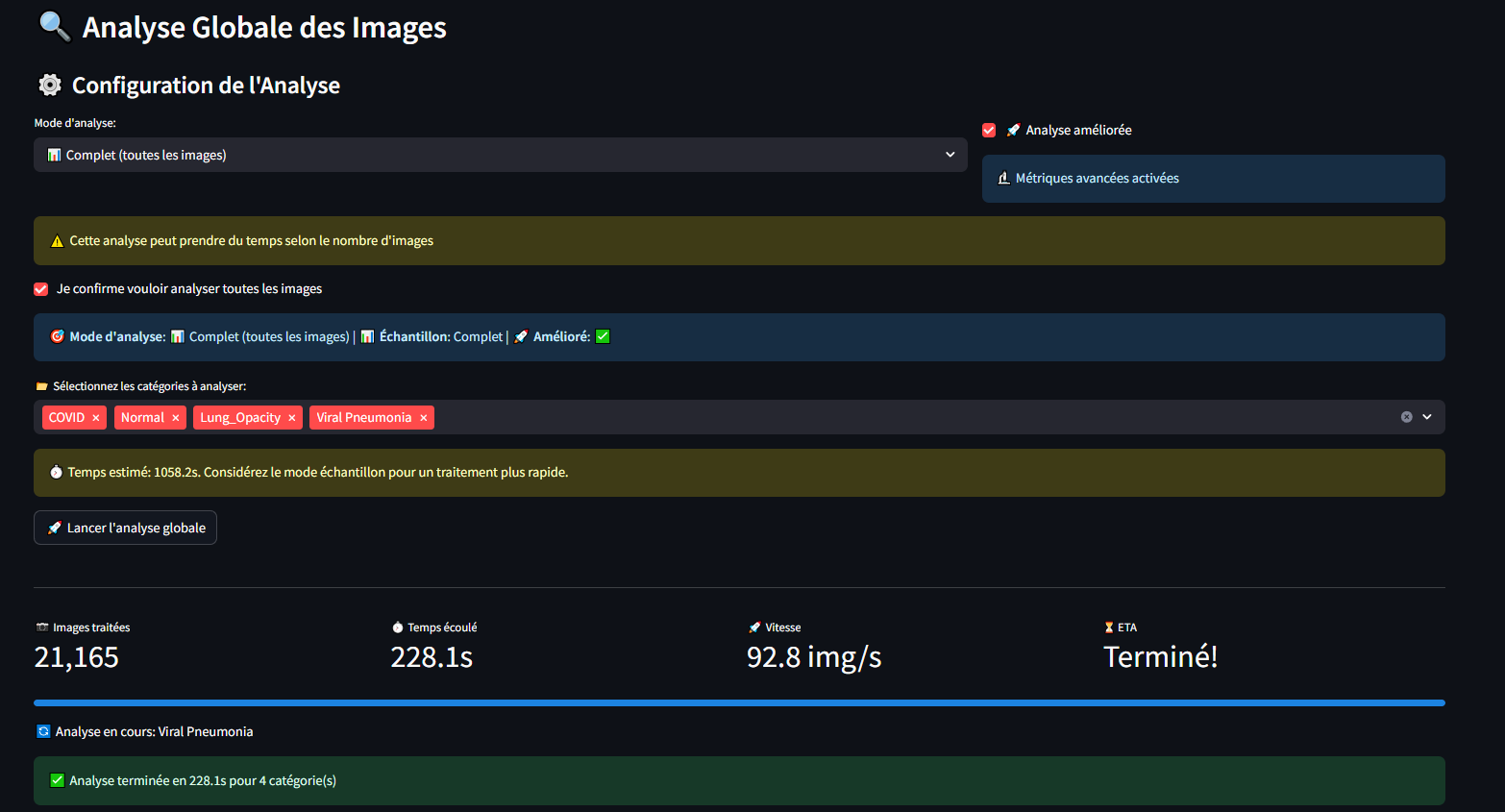
Il est ensuite possible d’afficher un panel détaillé avec toutes les informations essentielles de la fonction !

Ceci incluant le nom de la fonction, la doc string, le nombre de lignes, les arguments et la valeur de return.

 **Figure 09 :** Détail Fonctions Python Backend

## **Analyse Complète (Déséquilibrée) :**

Nous allons utiliser les fonctionnalités de l’interface afin de traiter et analyser nos données d’images.

**Figure 10 :** Menu Principal, sélection des données

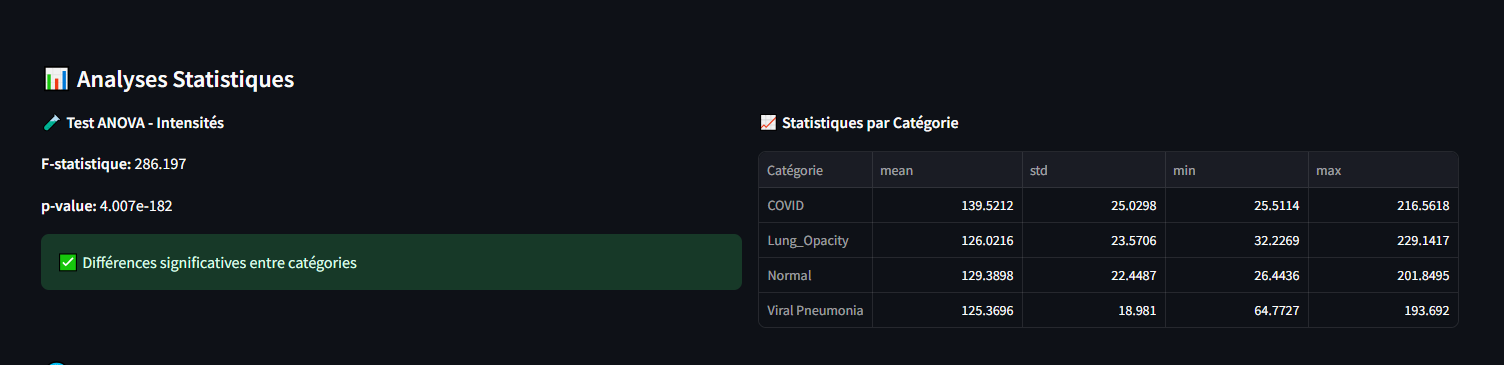
On lance une analyse sur l’ensemble des images.

**Figure 11 :** Radar Chart, Données/Métriques par Catégories

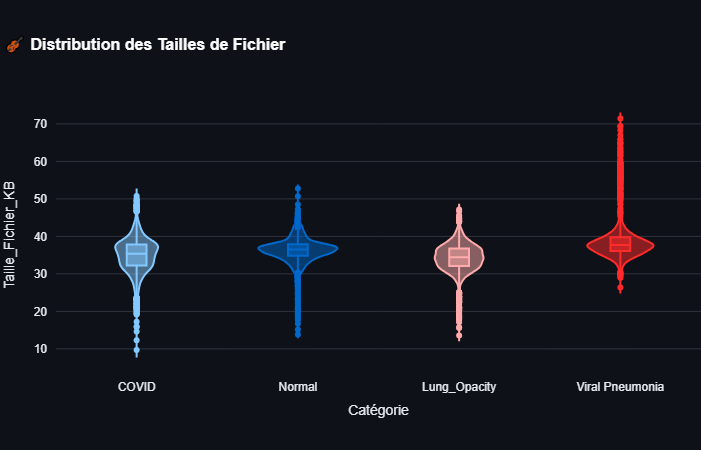
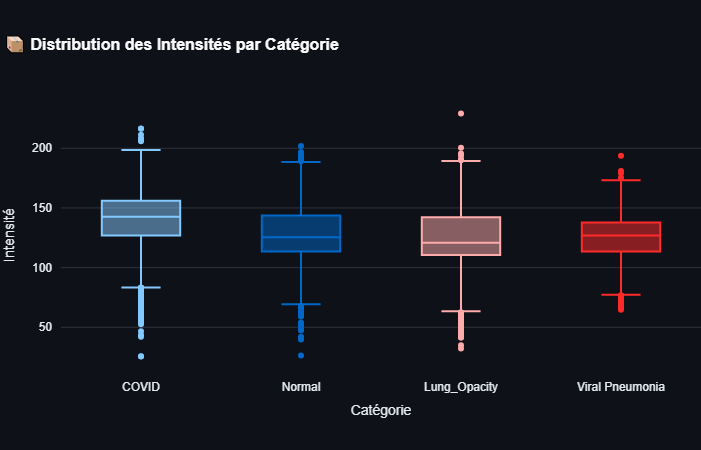
On peut déjà noter visuellement des domaines différents pour chaque catégorie.

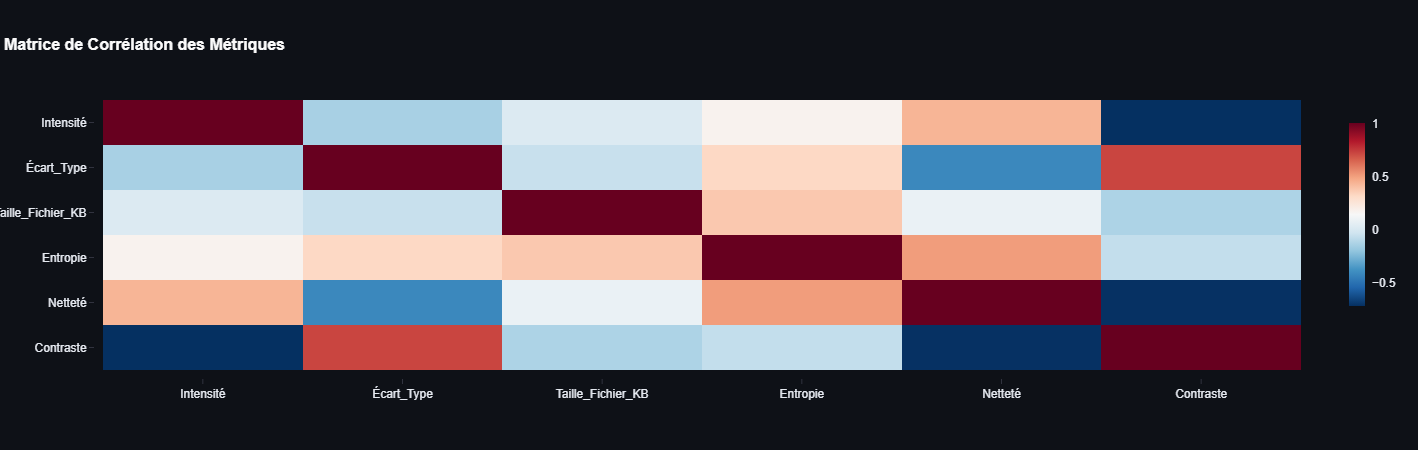
* **Luminosité / Intensité :** Moyenne des valeurs de pixels, reflète la clarté globale de l’image.
* **Contraste :** Écart-type des valeurs de pixels, mesure la variation autour de la moyenne.
* **Netteté :** Quantifie la présence de détails et de contours, via la fonction Laplacien.
* **Entropie :** Mesure la diversité des intensités, indique la richesse d’information.

Et l’on confirme ce résultat par un test ANOVA.

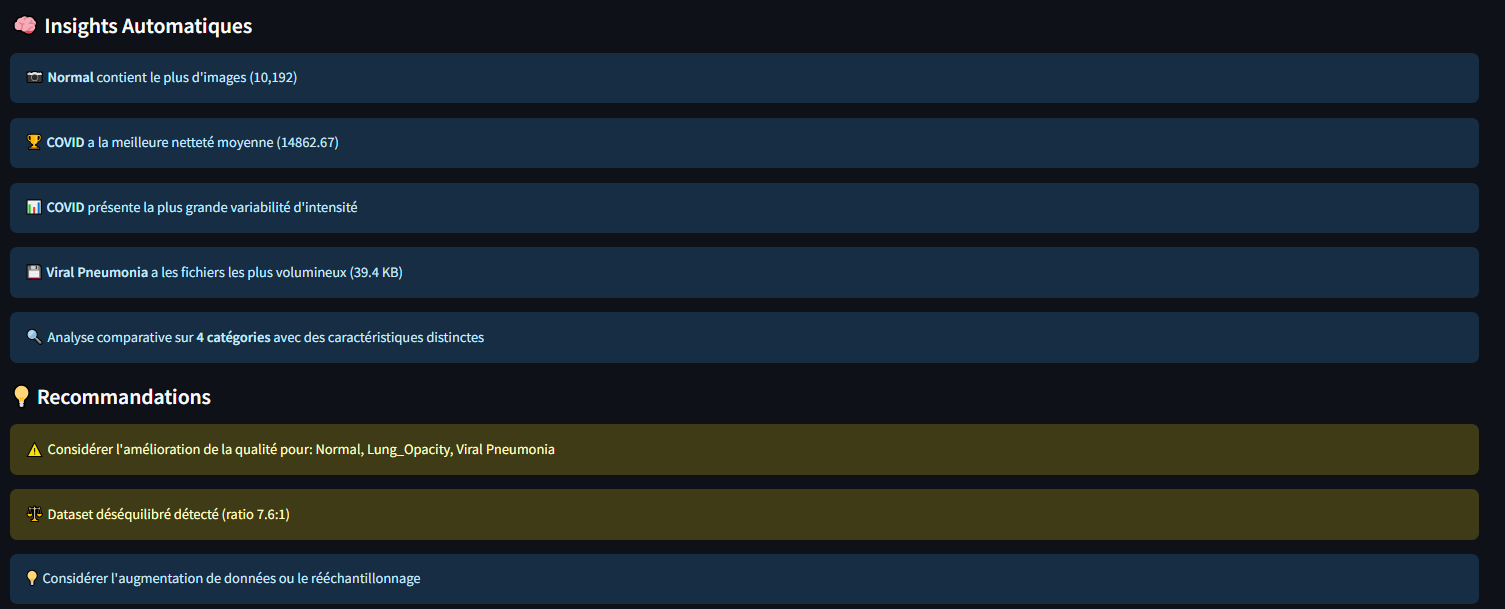


**Figure 12 :** Test Anova, différence entre Catégories

**Figure 13 :** Taille des fichiers par catégorie**Figure 14 :** Intensité par Catégorie

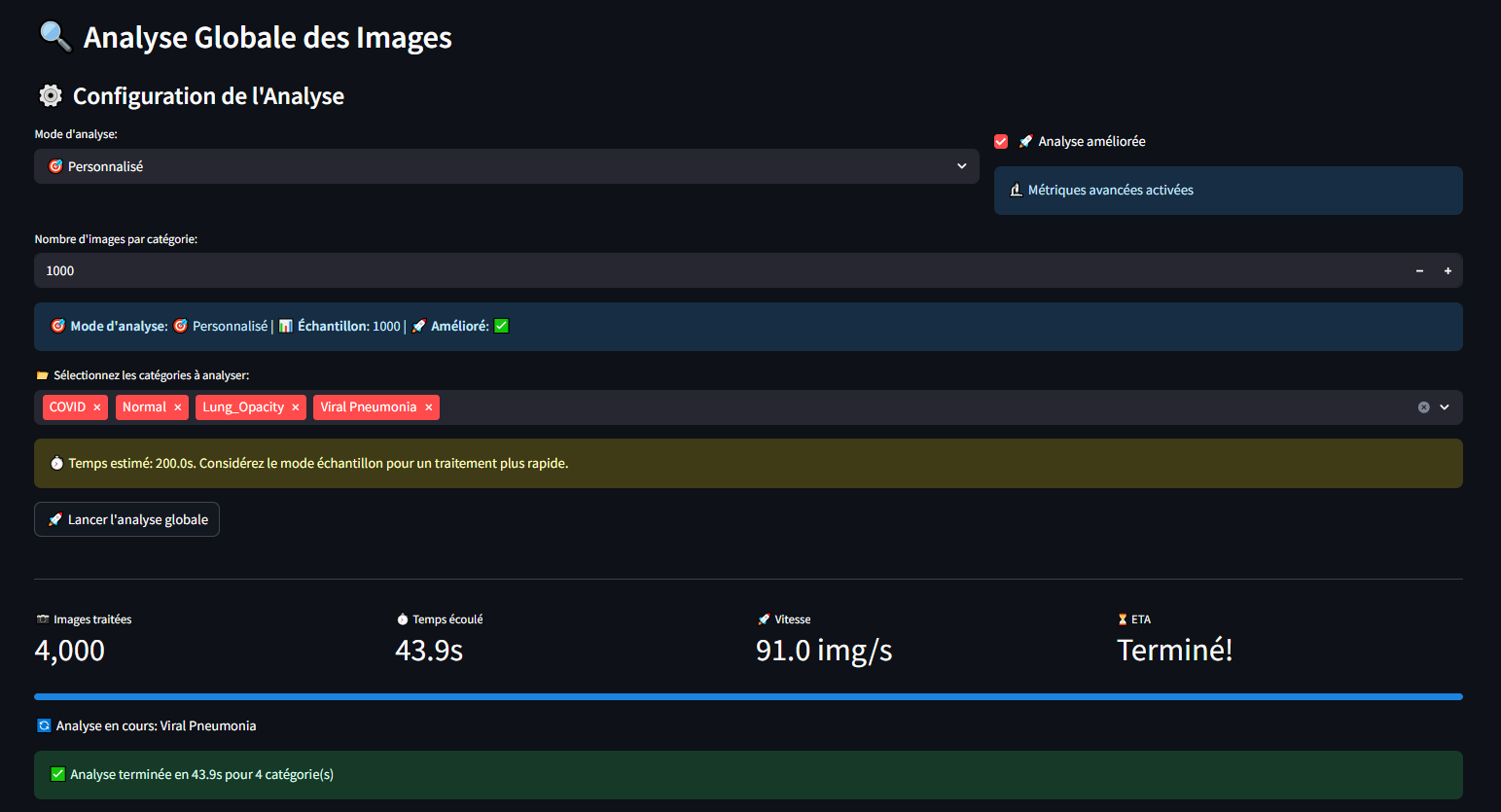
**Figure 15 :** Corrélation des métriques

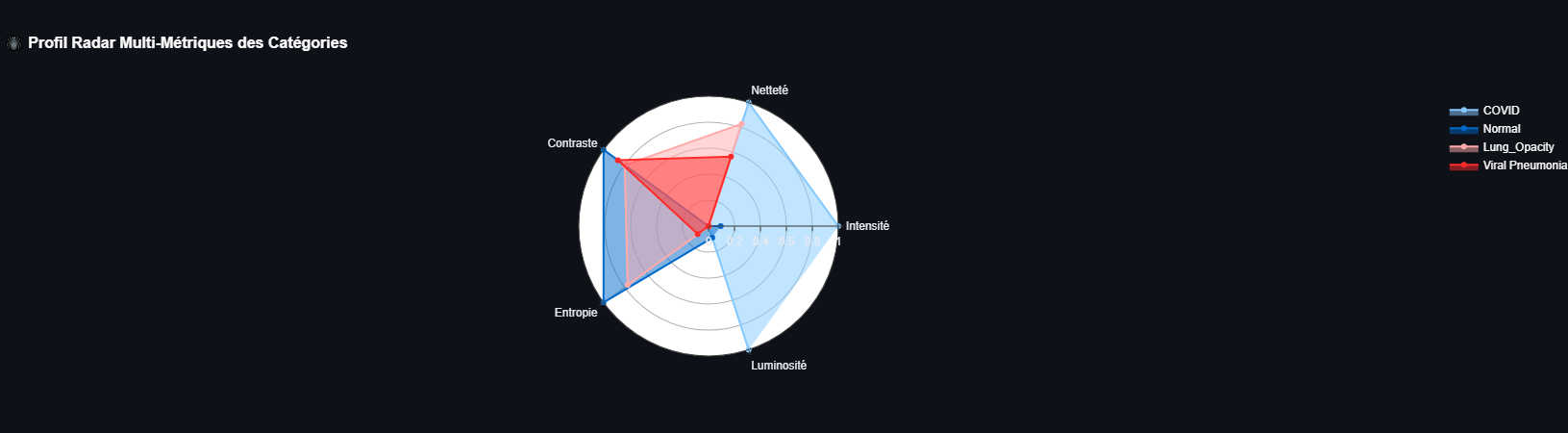
**Figure 16 :** Répartition des données dans un espace 3D

**Figure 17 :** Recommandations Automatiques

## **Analyse Equilibrée :**

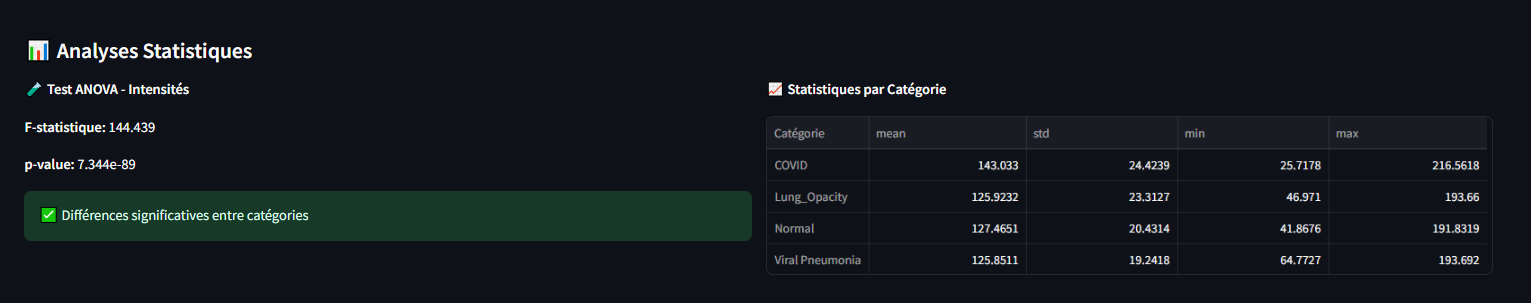
Suite à cette analyse, on se rend compte que les données ne sont pas équilibrées, on va donc retenter l’analyse avec un échantillon de 1000 pour chacune des catégories et voir les différences sur nos indicateurs.

**Figure 18 :** Menu Principal, sélection des données (équilibrés)**Figure 19 :** Menu Principal, Aperçu des données (équilibrés)

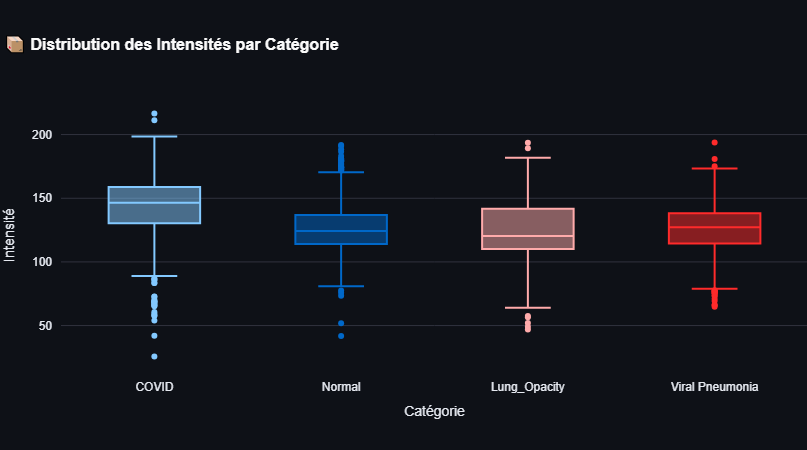


**Figure 20 :** Menu Principal, sélection des données (équilibrés)

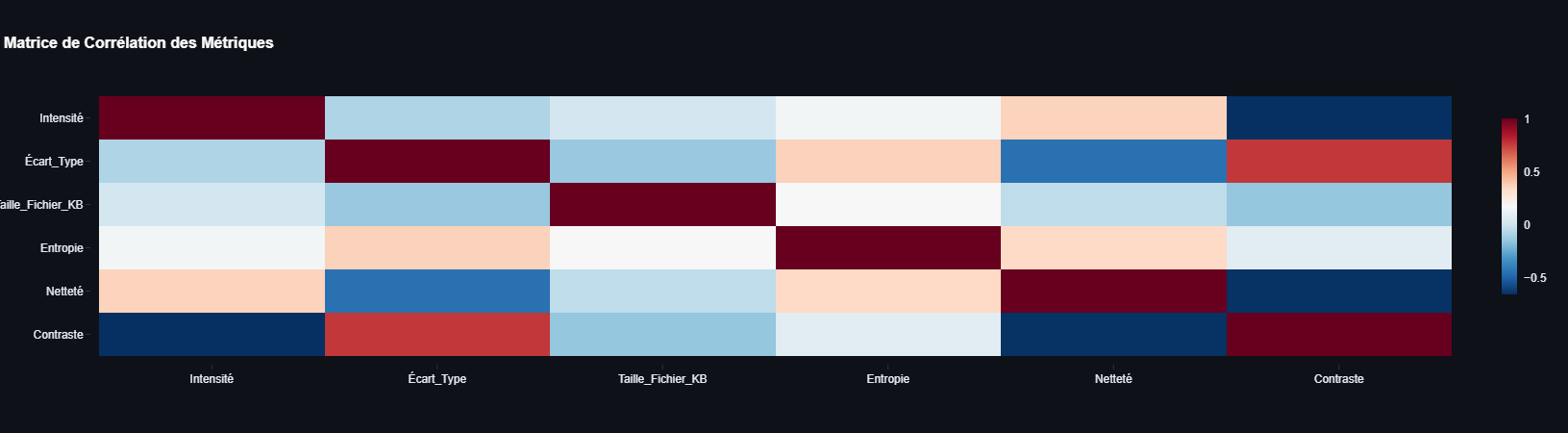
On peut noter sur cet ensemble plus équilibré que la représentation des domaines est différente par rapport à l’analyse précédente.

**Figure 21 :** Test Anova, Différences Catégories (équilibrés)

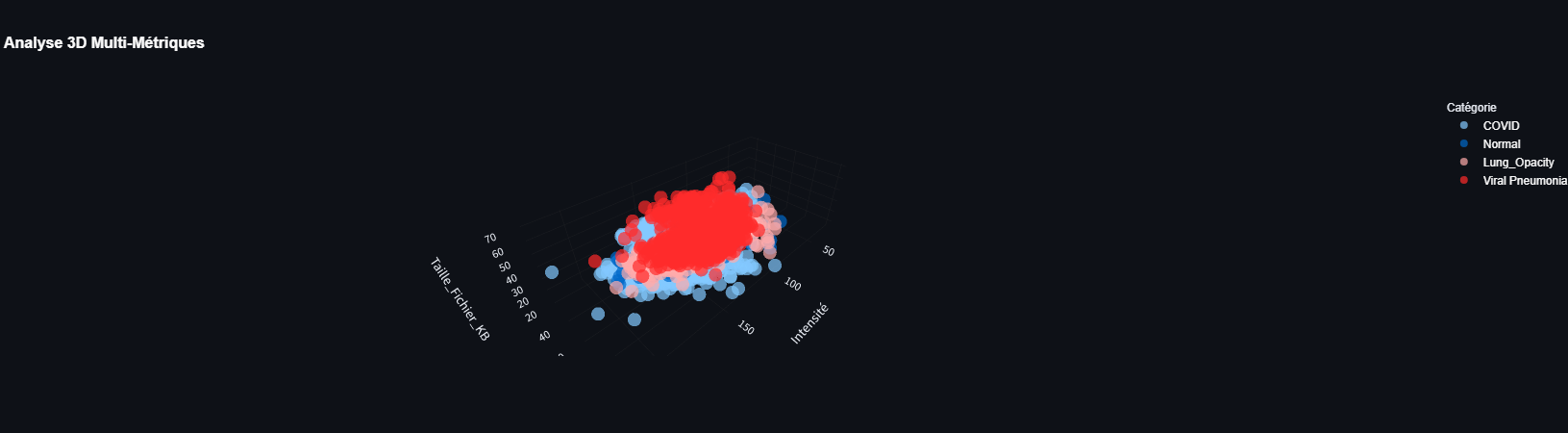
Les différences sont toujours significatives.

**Figure 22 :** Intensité par Catégories (équilibrés)

**Figure 23 :** Taille de fichiers par catégorie (équilibrés)

**Figure 24 :** Corrélation des métriques (équilibrés)

**Figure 25 :** Recommandations Automatiques (équilibrés)

**Figure 26 :** Représentation des données dans un espace 3D (équilibrés)

### **1.2.4 Génération images masquées + Statistiques :**

Cette fonctionnalité appelée par un notebook, permet de générer des images masquées à l’aide d’images et de masques (nos datas).

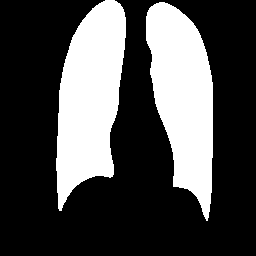
Pendant cette opération, des statistiques sont également évalués et représentées.

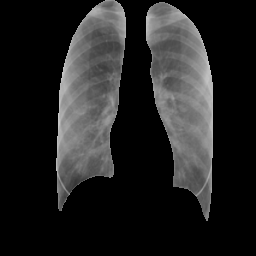
Voici les statistiques du masquage sur l’ensemble des données, avec un paramètre :

dsize = (256x256), ce qui correspond à une image de 256x256 pixels.

## **Exemple Génération Image Masqué :**

La fonction permet en lui donnant une image et un masque, de l’appliquer pour générer une image masquée, avec une possibilité de redimensionnement de l’image. (Figure 27)





**Figure 27 :** Exemple, Masquage des images

# **Etape 02: Pre-processing et features engineering**

Suite à l’étape de datavisualisation menée sur le jeu de données, une analyse exploratoire approfondie nous a permis d’identifier les caractéristiques clés et les problématiques spécifiques à adresser lors du préprocessing. Les visualisations ont mis en évidence la nécessité d’un traitement homogène des images pour faciliter l’analyse ultérieure. En conséquence, la première opération consistera à convertir l’ensemble des images initialement au format RGB en niveaux de gris. Cette conversion permet d’éliminer l’information chromatique et de se concentrer sur l’intensité lumineuse, simplifiant le cadre d’apprentissage et renforçant la cohérence inter-échantillons.

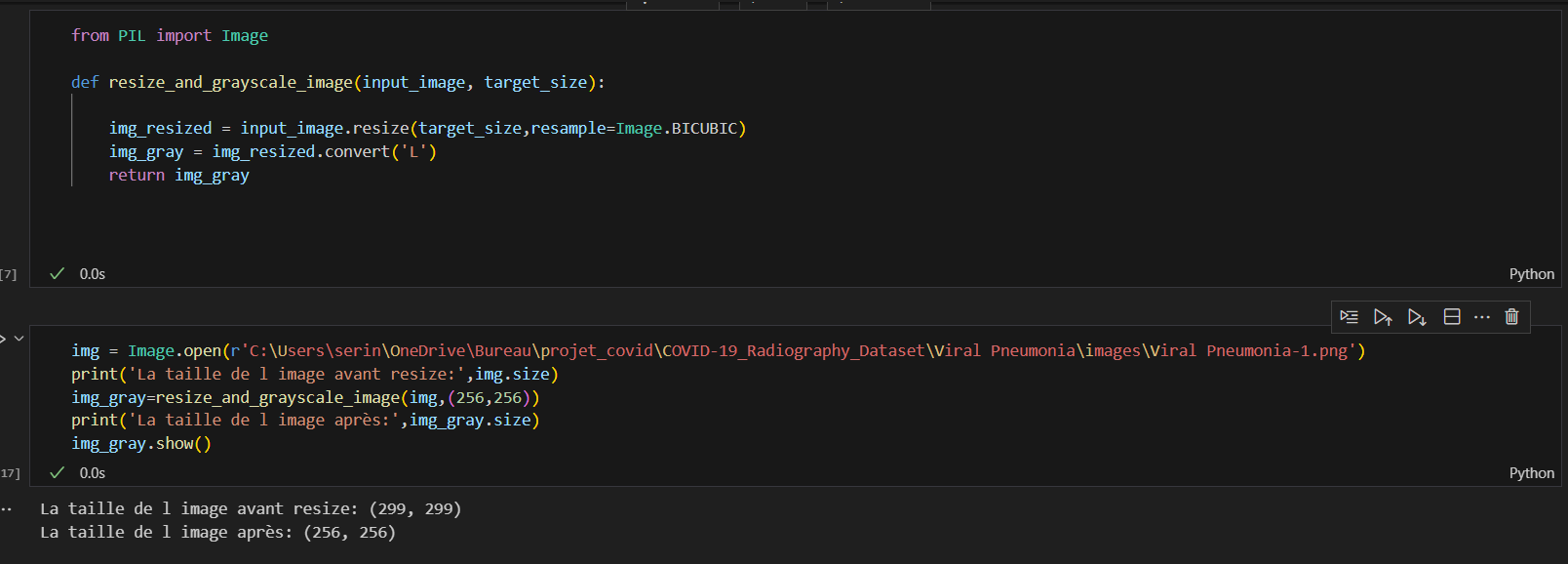
Par ailleurs, l’étude des classes a révélé un déséquilibre important entre les catégories « pneumonia » et « normal ». Pour pallier ce problème et éviter les biais d’apprentissage, nous appliquerons des stratégies d’échantillonnage adaptées : un sur-échantillonnage sera réalisé pour la classe minoritaire « pneumonia » afin d’accroître sa représentativité dans le corpus, tandis qu’un sous-échantillonnage viendra réduire la taille de la classe majoritaire « normal », poursuivant ainsi l’objectif d’obtenir un jeu de données plus équilibré et approprié à l’entraînement de modèles robustes.

## **2.1 Redimensionnement et conversion en niveaux de gris :**

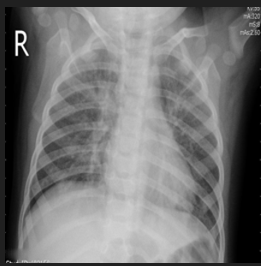
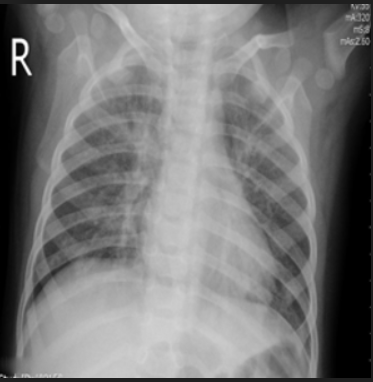
Dans la phase initiale du pré-processing des données, nous appliquons un redimensionnement des images dont la dimension initiale est de 299 pixels, pour les réduire à une taille uniforme de 256 pixels. Ce choix, vise à garantir une cohérence des entrées du modèle tout en facilitant le traitement par lot et en réduisant la charge computationnelle. Ce redimensionnement est réalisé en utilisant des techniques d’interpolation adaptées, telles que l’interpolation bicubique, afin de préserver au mieux la qualité et les détails pertinents des images.

Par la suite, la conversion de l’ensemble des images du format RGB vers une représentation en niveaux de gris est effectuée. Cette transformation réduit la dimension d’entrée des données de trois canaux à un seul, diminuant ainsi la complexité du modèle tout en conservant l’essentiel de l’information structurelle pertinente, particulièrement dans le contexte d’images médicales où la couleur est souvent secondaire. Cette étape est particulièrement judicieuse dans un cadre de classification, car elle simplifie le signal d’entrée, réduit le bruit potentiel lié à la variation chromatique, et permet d’optimiser les performances des algorithmes d’apprentissage en concentrant l’analyse sur les contrastes et textures essentielles à la discrimination des classes.

Le code utilisé est dans la figure ci-dessous:(**Figure 28**)

  **Figure 28 :** le code utilisé pour le redimensionnement et la conversion en niveau de gris

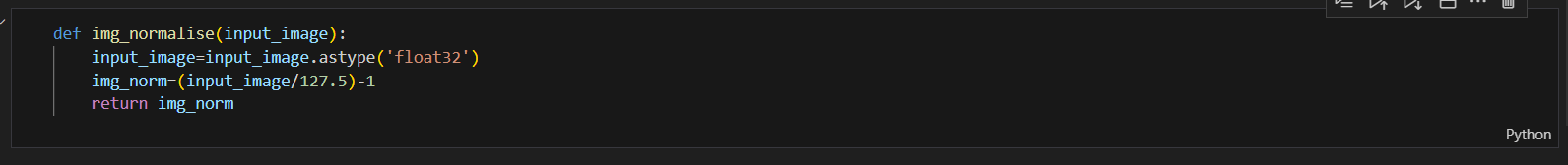
Un exemple des images de la classe pneumonie obtenue après conversion en niveau de gris et redimensionnement est représenté dans la figure (Figure 30) suivante :



**Figure 29 :** Exemple d’une image de la classe pneumonie avant et après redimensionnement et conversion

## **2.2 Normalisation des pixels :**

Après redimensionnement, la normalisation des valeurs de pixels est essentielle pour aider les modèles d’apprentissage profond à converger rapidement, dans notre cas nous avons choisie de normaliser les pixels dans une plage [-1,1], le code utiliser est représenter dans la figure suivante : (Figure 31)



**Figure 30 :** Fonction de normalisation

## **2.3 Augmentation data**

Après l’étude menée lors de l’étape 1, nous avons constaté un déséquilibre important au niveau des classes de notre jeu de données. Pour remédier à cette situation, nous appliquerons un sous-échantillonnage afin de réduire la taille, et un sur-échantillonnage à la classe pour augmenter le nombre d’exemples et ainsi atteindre un équilibre plus satisfaisant entre les classes. Ces méthodes permettront d’améliorer la performance et la robustesse de notre modèle en limitant les biais liés à une représentation inégale des classes.

### **2.3.1 Sous échantillonnage : (code en annexe 02)**

Le sous-échantillonnage consiste à réduire le nombre d’exemples de la classe majoritaire dans un jeu de données afin d’équilibrer la distribution des classes. Cette méthode implique généralement de supprimer aléatoirement des exemples de la classe majoritaire jusqu’à ce que sa taille soit comparable à celle de la classe minoritaire. L’objectif est d’éviter que la classe dominante ne biaise la performance du modèle, tout en cherchant à conserver la diversité des données restantes. [4]

### **2.3.2 Sur échantillonnage : (code en annexe 04)**

Le sur-échantillonnage consiste à augmenter le nombre d’exemples de la classe minoritaire, soit en dupliquant aléatoirement des exemples existants, soit en générant de nouveaux exemples synthétiques. Ce procédé permet d’équilibrer la taille des classes et de renforcer la capacité du modèle à apprendre les caractéristiques propres à la classe minoritaire, limitant ainsi les problèmes liés à un déséquilibre de classes lors de l’entraînement. [5]

Lors du prétraitement des données, nous avons initialement chargé 3616 images pour la classe COVID, 10192 images pour la classe NORMAL, 1345 images pour la classe VIRAL et 6012 images pour la classe LUNG. Afin d'équilibrer le nombre d'images par classe, une stratégie combinant sous-échantillonnage (undersampling) et sur-échantillonnage (oversampling) a été appliquée : la classe VIRAL, initialement sous-représentée, a été sur-échantillonnée par augmentation de données pour atteindre 3000 images, tandis que les classes NORMAL, COVID et LUNG serront sous-échantillonnées afin d’obtenir un ensemble de données plus équilibré. Cette démarche vise à limiter les biais liés à un déséquilibre entre classes lors de l’entraînement du modèle.

Un exemple des résultats obtenus par la data augmentation en utilisant le sur échantillonnage est représenté dans la Figure 31.



**Figure 31:** Exemple d’images obtenus avec le sur échantillonnage

**Etape 03 : Modélisation**

## **3.1 Model Baseline**

Un modèle baseline désigne un modèle simple ou rudimentaire utilisé en machine learning pour servir de point de comparaison aux modèles plus complexes développés par la suite. Son objectif principal n’est pas d’obtenir la meilleure précision possible, mais d’établir un niveau de performance minimal que tout modèle plus sophistiqué doit dépasser pour être jugé pertinent. [6]

* Un modèle baseline en classification peut consister à prédire systématiquement la classe majoritaire du jeu de données.
* En régression, il s’agira par exemple de prédire la valeur moyenne ou médiane de la variable cible, quelle que soit l’entrée.

Les modèles baseline ne cherchent pas la performance, mais servent à évaluer la difficulté intrinsèque du problème.

Exemples de modèles baseline :

* Pour classification : prédire la classe majoritaire, prédire au hasard, DummyClassifier.
* Pour régression : prédire la moyenne/médiane, DummyRegressor.

Nous avons choisi de tester les méthodes de machines learning suivantes : SVM, KNN et Forest Random.

### **3.1.1 Modele Machine learning:**

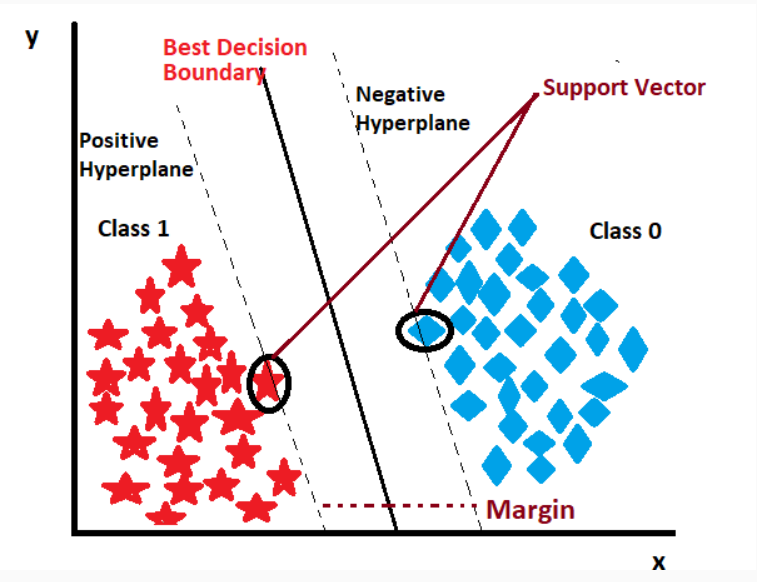
**SVM (Support Vector Machine):**

Un SVM (Support Vector Machine) est un algorithme supervisé de classification et de régression. (Figure 32)

Il cherche à tracer une frontière (hyperplan) qui sépare au mieux les classes dans un espace de caractéristiques, en maximisant la "marge" entre les points de chaque classe les plus proches de la frontière (les "vecteurs support").

C’est donc un modèle robuste, notamment pour les données de grande dimension, qui peut aussi intégrer des fonctions noyau ("kernel") pour gérer la non-linéarité : il projette les données dans un espace de plus grande dimension pour trouver une séparation optimale.

Utilisé en binaire ou multiclasse, il est réputé pour sa précision sur des bases bien structurées.[7]



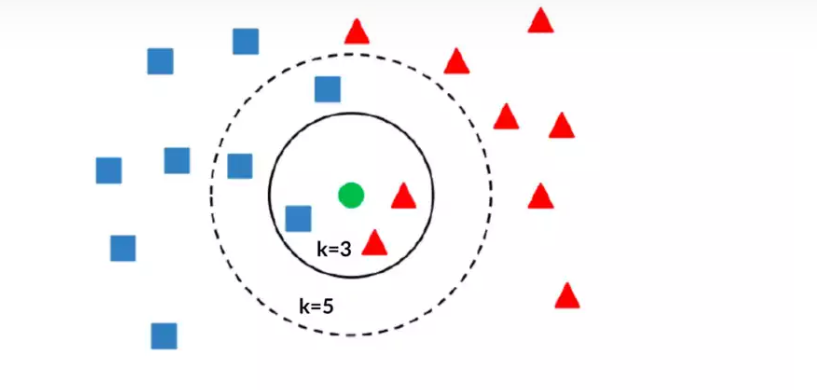
**Figure 32 :** Principe de fonctionnement du SVM

**K-Nearest Neighbors (KNN) :**

Le KNN ("K plus proches voisins") est un algorithme non paramétrique supervisé.

Son principe : pour une nouvelle donnée à classer, il regarde les K points les plus proches dans le jeu d’entraînement (selon une mesure de distance, souvent euclidienne).

Il attribue alors la classe la plus fréquente parmi ces voisins ("majorité") en classification, ou la moyenne pour la régression. (Figure 33) [7]



**Figure 33 :** Principe de fonctionnement des KNN

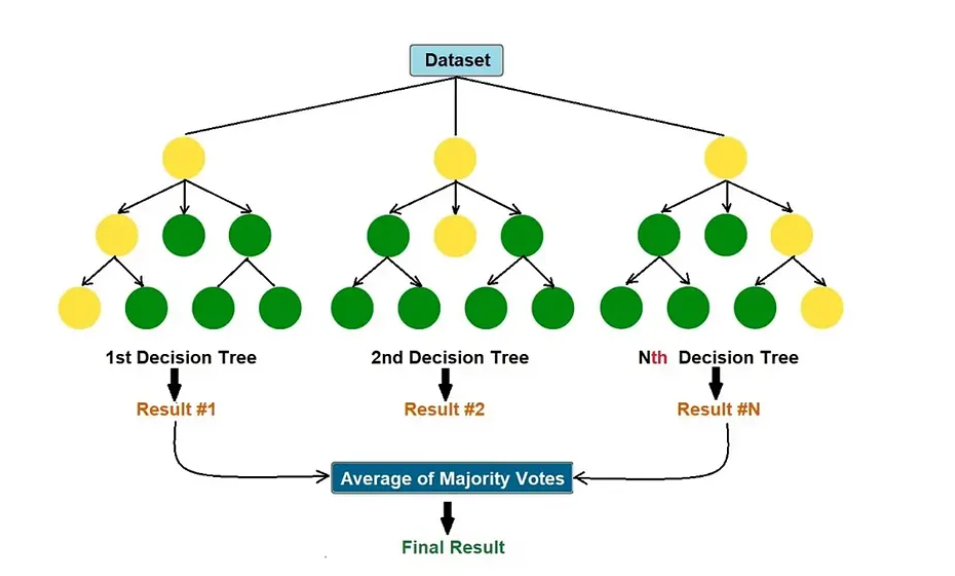
**Random Forest (Forêt d'Arbres Décisionnels) :**

Le Random Forest est un algorithme "ensembliste" basé sur les arbres de décision.Il construit une multitude d'arbres de décision sur des sous-échantillons aléatoires des données (méthode du "bagging").

Chaque arbre fait une prédiction, et dans le cas de la classification, la classe majoritaire parmi tous les arbres devient la prédiction finale ("vote majoritaire"). En régression, c’est la moyenne des prédictions.

Les arbres sont construits avec des sous-ensembles aléatoires de variables/features et de données, ce qui réduit le sur-apprentissage et augmente la robustesse.

Le modèle est très performant pour les données tabulaires et mixte (catégorielles / numériques), mais peut-être moins adapté à des données très éparses ou avec beaucoup de valeurs manquantes. (Figure 34) [7]

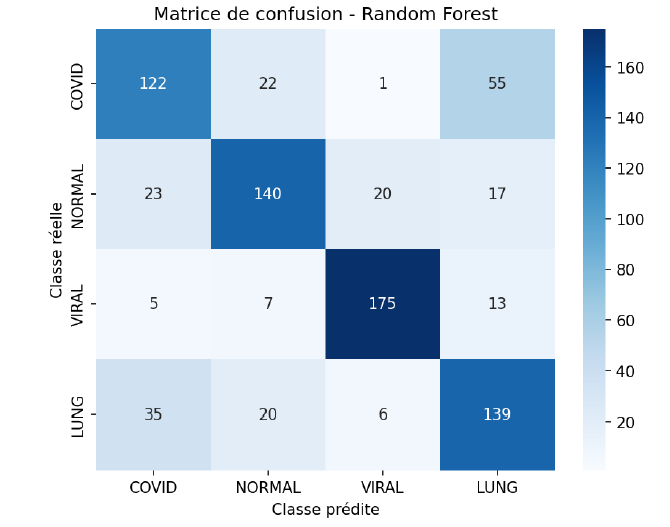
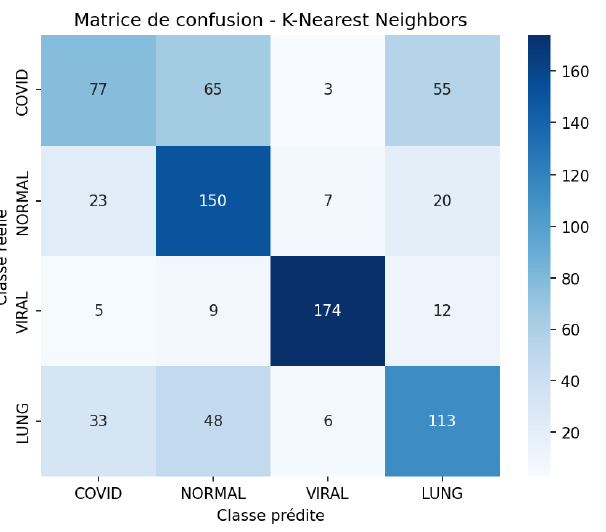
**Figure 34 :** Principe de fonctionnement Random Forest

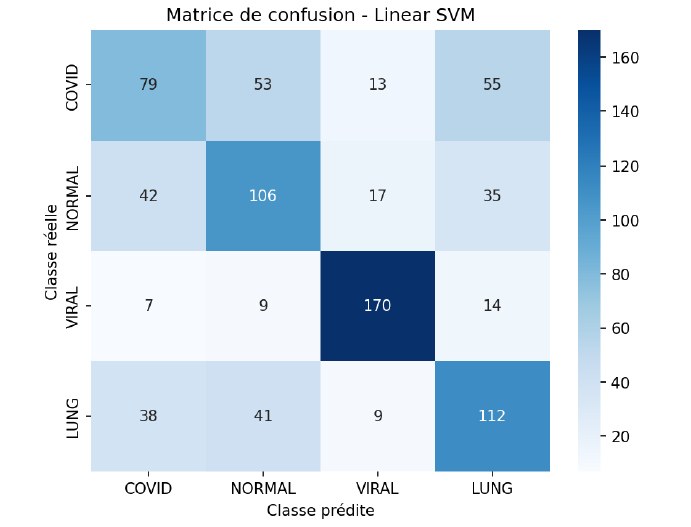
**Avantages et Inconvénients :**

* **KNN :** très simple, mais peu efficace sur des volumes importants ou en présence de bruit.
* **SVM :** performant et robuste sur petites/moyennes dimensions, mais coûteux et difficile à régler.
* **Random Forest :** généraliste et précis, bien adapté aux données complexes, mais lourd en calcul et moins interprétable.[7]

**Résultats et Interprétation :**

Les résultats de la matrice de confusion obtenus avec les 03 modèles de machine learning sur 200 données d’entrainement de chaque classe sont représentés dans la figure ci-dessous : (Figure 35)





**Figure 35 :** Résultats de la matrice de confusion pour les trois modèles SVM, K-NN et Random Forest

D’après les résultats obtenus on a :

|  |  |  |  |
| --- | --- | --- | --- |
| **Modèle** | **Points forts** | **Points faibles** | **Performance globale** |
| **Random Forest** | Bonne précision, surtout sur NORMAL et VIRAL | Confusion COVID↔LUNG | **Meilleur compromis** |
| **Linear SVM** | Bonne généralisation, VIRAL bien reconnu | Confusion LUNG ↔ COVID et NORMAL ↔ COVID | **Moins performant** |
| **KNN** | SIMPLE, NORMAL bien reconnu | Confusions massives, surtout COVID↔NORMAL et LUNG ↔ NORMAL | Moyenne |

Le Random Forest est le modèle le plus performant et le plus équilibré pour cette tâche.  
 Il présente la meilleure capacité de généralisation, notamment pour les classes NORMAL et VIRAL, malgré une confusion entre COVID et LUNG, qu’on pourrait réduire par une meilleure sélection de caractéristiques ou un ensemble de données plus diversifié ou en utilisant des méthodes d’optimisations.

## **3.2 Deep Learning :**

**3.2.1 Modèle base line finetuning :**

Le fine-tuning est une technique d'apprentissage automatique qui consiste à ajuster un modèle pré-entraîné (par exemple sur ImageNet pour la vision ou sur un gros corpus de textes pour le NLP) afin de l'adapter à une tâche spécifique avec un nouveau jeu de données plus restreint.[8] Dans notre projet nous avons choisi d’utilisé InceptionV3.

Le modèle baseline InceptionV3 est un réseau de neurones à convolution profond de 48 couches comportant des modules « Inception » qui exécutent plusieurs opérations convolutives de tailles différentes en parallèle, puis concatènent leurs résultats, conçu par Google, très utilisé en vision par ordinateur pour la classification d’images. Il sert de point de départ (baseline) lorsqu’on applique un transfert d’apprentissage.

**Avantage :**

* Précision élevée pour la classification d’images grâce à la profondeur et à la structure modulaire du réseau.
* Pré-entraînement sur ImageNet : permet d’obtenir immédiatement de bonnes performances sur des jeux de données où les images ont des similarités avec ImageNet.
* Baseline : sert de référence minimale de performance avant toute adaptation personnalisée. [9]

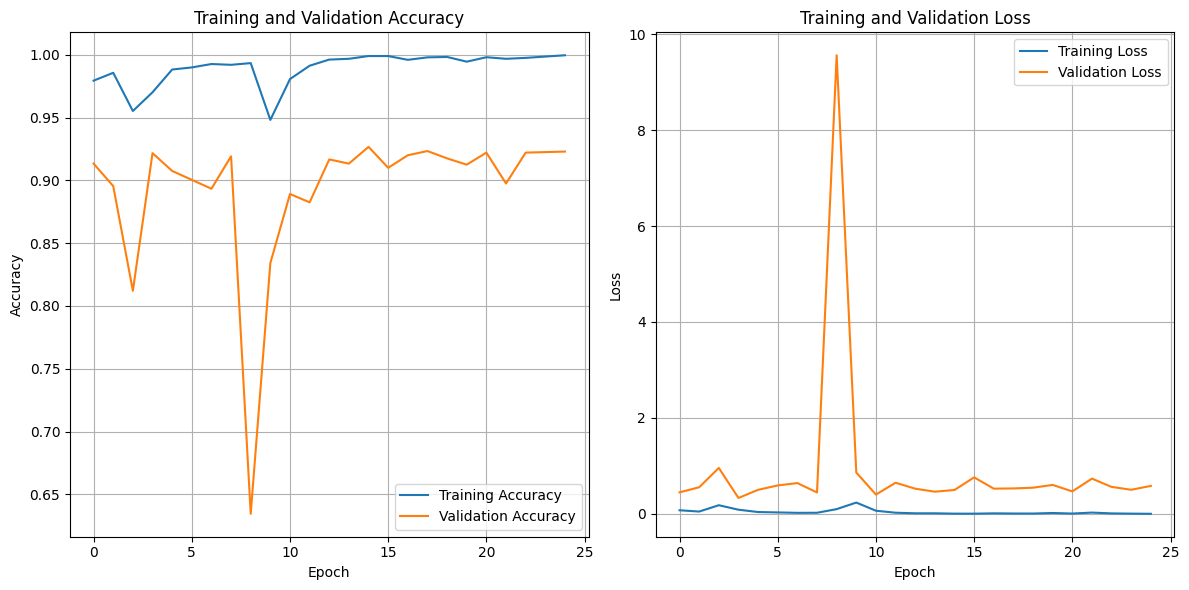
**3.2.2 Entrainement du modèle de deep learning:**

Pour la classification multiclasses des images radiographiques en COVID-19, pneumonie lung et normal, nous avons utilisé le modèle InceptionV3 pré-entraîné sur ImageNet comme base d’extraction de caractéristiques. Pour adapter ce modèle à notre tâche spécifique, nous avons employé le modèle séquentiel de Keras, qui permet de créer un réseau de neurones en empilant des couches les unes après les autres. Nous avons ajouté sur la base InceptionV3 plusieurs couches personnalisées : une couche d’aplatissement (Flatten), une couche dense avec activation ReLU, une couche Dropout pour éviter le surapprentissage, puis une couche dense finale avec activation softmax correspondant au nombre de classes. Le modèle a ensuite été compilé avec l’optimiseur Adam, la fonction de perte adaptée aux labels entiers (sparse categorical crossentropy) car on traite un cas de classification multiclasse et la métrique d’évaluation précision (La précision est une métrique d’évaluation utilisée en classification, notamment en classification d’images médicales, qui mesure la proportion d’exemples correctement prédits parmi l’ensemble des exemples classés positivement par le modèle. Formellement, la précision se calcule comme le nombre de vrais positifs divisé par la somme des vrais positifs et des faux positifs, En d’autres termes, la précision quantifie la capacité du modèle à ne pas générer de faux positifs).

De plus, nous avons intégré le callback ReduceLROnPlateau, qui ajuste automatiquement le taux d’apprentissage en fonction de la stagnation de la métrique de validation. Ce callback réduit le learning rate lorsque la performance cesse de s'améliorer, aidant ainsi le modèle à converger plus finement et à sortir des plateaux d’apprentissage.

Cette approche simple et modulaire combine ainsi la puissance de généralisation d’InceptionV3 avec une adaptation personnalisée et un contrôle dynamique du taux d’apprentissage, optimisant efficacement notre problème médical.

Notre modèle a été entrainé sur un nombre de données de 3000 données de chaque classe et avec un nombre d’epochs de 25 epochs les résultats des courbes d’accuracy et de loss obtenus sont représentés dans la figure36.



**Figure 36 :** Courbe de loss et d’accuracy

## **Interprétation :**

**Courbe de précision (Accuracy) :**

* La courbe de précision d’entraînement demeure très élevée tout au long des époques, se stabilisant autour de 0.98 à 1.00. Cela indique que le modèle apprend efficacement sur les données d’entraînement.
* La précision sur la validation est généralement stable autour de 0.90, avec de légères fluctuations et un point de chute net vers la 8ᵉ époque avant un retour rapide vers des valeurs élevées.
* Cette différence entre la précision d’entraînement et de validation, notamment le pic de baisse, peut signaler une difficulté du modèle à généraliser sur certains lots de validation (batch) ou une instabilité temporaire liée à la répartition des données.

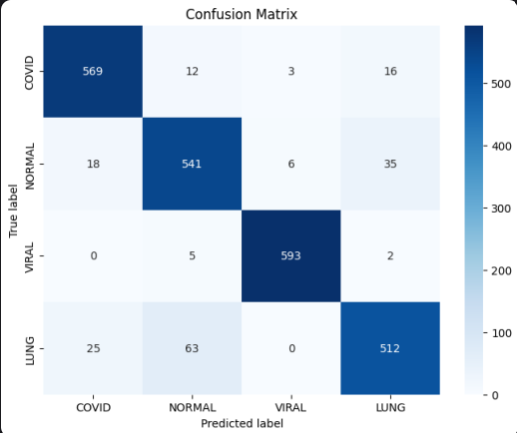
**Courbe de la fonction de perte (Loss) :**

* La perte d’entraînement reste constante et très faible, ce qui montre une optimisation maîtrisée du modèle sur les données apprises.
* La perte de validation présente plusieurs fluctuations et un pic important à la 8ᵉ époque, ce qui correspond à la baisse de précision observée sur la même époque.
* Après ce pic, la courbe retrouve sa tendance initiale. Ce comportement peut être lié à l’arrivée de batchs atypiques en validation, un bruit dans les données, ou à une étape critique de régularisation du modèle.

**Conclusion :**

* Le modèle apprend très bien et atteint une précision très haute sur l’entraînement.
* La précision en validation reste élevée, mais le modèle montre une légère tendance au surapprentissage (overfitting), visible par la différence entre les deux courbes et les pics de perte/acc.
* Pour améliorer la robustesse, il pourrait être pertinent d’effectuer davantage d’optimisations : data augmentation (diversifier les données), ou tuning de l’hyperparamètre de régularisation.

## Nous avons aussi calculé la matrice de confusion,les résultats de la matrice de confusion obtenus pour le modèle de deep learning sont représentés dans la figure 38.



## 

## **Figure 38 :** les résultats de la matrice de confusion pour le modèle de deep learning

## La matrice de confusion générée par le modèle de deep learning montrent que notre modèle est très performant et qu’il arrive a bien identifié les différentes classes : COVID, pneumonia, virale et état normal.

## **3.3 Optimisation :**

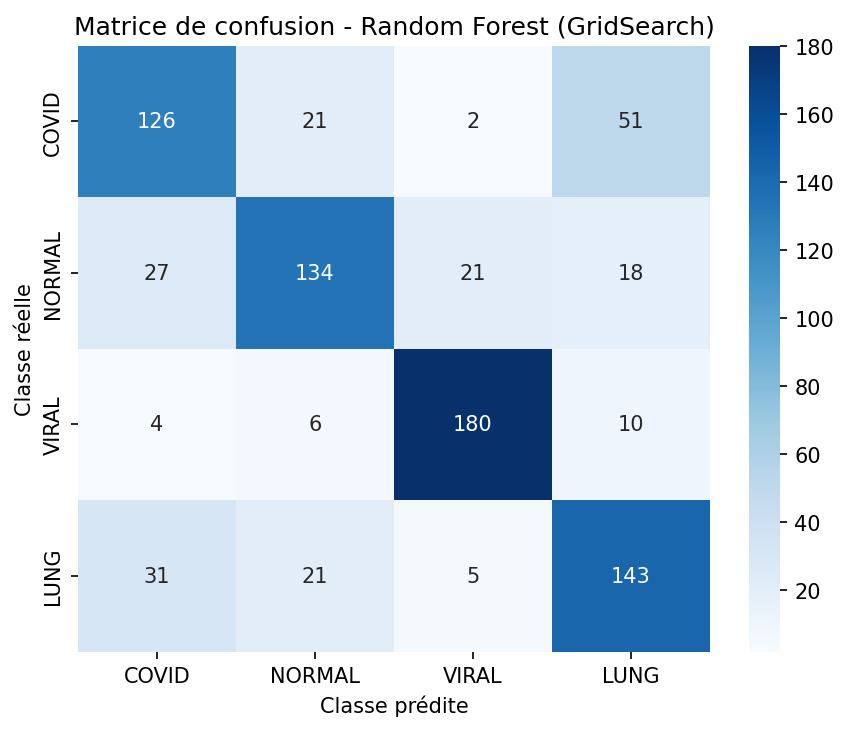
Pour cette partie, nous avons choisi d’optimiser à la fois le modèle le plus performant, à savoir le Random Forest, et le modèle le moins performant, le SVM, en nous basant sur les résultats précédemment obtenus. Cette décision vise à améliorer davantage le modèle qui a déjà démontré une forte efficacité tout en cherchant à renforcer les performances du modèle présentant des résultats moins satisfaisants, afin d’avoir une analyse comparative plus complète.

### **3.3.1 Grid search:**

La grid search (ou recherche sur grille) est une méthode d’optimisation utilisée en machine learning pour déterminer la meilleure combinaison d’hyperparamètres d’un modèle. Cette technique consiste à définir un espace de recherche constitué de toutes les valeurs possibles pour chaque hyperparamètre, puis à tester de manière systématique toutes les combinaisons possibles afin de trouver celle qui donne les meilleurs résultats selon un critère de performance (par exemple, l’exactitude ou la précision du modèle).[10]

**Résultats et interprétation :**

Nous avons testé la méthode grid search sur les 02 modèles de machine learning SVM et Random Forest sur 200 données d’entrainement le résultat de la matrice de confusion obtenu est représenté dans la figure 39.



**Figure 39 :** Résultats de la matrice de confusion pour les modèles SVM et Random Forest avec Grid search

Après avoir utilisé la méthode grid search, nousavons constaté une amélioration des performances pour certaines classes avec les deux méthodes, SVM et Random Forest. Cependant, il persiste une certaine confusion dans la prédiction entre certaines classes, ce qui montre que malgré les améliorations, des erreurs d’identification entre catégories proches subsistent. Cette observation souligne la complexité de la classification multi-classes dans notre contexte médical et met en avant la nécessité d’utiliser une autre approche que les modèles de machine learning et opter pour un modèle de deep learning.

**3.3.2 XGBoost Classifier :**

Le XGBoost Classifier (eXtreme Gradient Boosting) est un algorithme d’apprentissage supervisé basé sur l’assemblage d’arbres de décision, utilisant la technique du gradient boosting. Il consiste à entraîner plusieurs arbres faibles (weak learners) de façon séquentielle, chaque nouvel arbre cherchant à corriger les erreurs des arbres précédents en se concentrant sur les résidus, c’est-à-dire les exemples mal prédits.

XGBoost se distingue par sa rapidité, son efficacité sur de grands jeux de données et ses nombreuses optimisations (parallélisation, gestion efficace de la mémoire, régularisation pour limiter le surapprentissage). Il est reconnu pour offrir d’excellentes performances en classification et en régression, notamment sur des données structurées ou tabulaires. L’algorithme a dominé de nombreuses compétitions Kaggle grâce à sa robustesse, sa capacité à gérer des ensembles de données volumineux et sa flexibilité d’utilisation.[11]

**Résultats et interprétation :**

Nous avons testé la méthode XGBoost sur 200 données d’entrainement le résultat de la matrice de confusion obtenu est représenté dans la figure 40.



**Figure 40 :** Résultats de la matrice de

confusion pour le XGBoost Classifier

Le XGBoost Classifier montre de bonne performance globale pour la détection des cas lung et viral, mais nécessite encore un raffinement pour les classes covid et normal.

**3.4 Interprétabilité :**

**3.4.1 Grad CAM: (Gradient-weighted Class Activation Mapping)**

Est une méthode d'explicabilité en deep learning qui permet de visualiser les zones importantes d'une image ayant conduit un réseau de neurones convolutifs (CNN) à faire une prédiction. Cette technique génère une carte thermique ("heatmap") indiquant quelles parties de l'image ont le plus influencé la décision du modèle.

Le procédé s'appuie sur le calcul des gradients de la sortie liée à une classe cible par rapport aux cartes de caractéristiques (feature maps) produites par la dernière couche convolutionnelle du réseau. Ces gradients indiquent l'importance de chaque région de l'image pour la prédiction de la classe, et sont combinés pour créer la carte d'activation localisée.

Grad-CAM est utile pour comprendre et interpréter visuellement les décisions d'un modèle, identifier des biais éventuels, et s'assurer que le réseau se concentre bien sur des éléments pertinents de l'image au lieu de zones non informatives.[12]

**Résultats et interprétation :**

**3.4.2 SHAP : (SHapley Additive exPlanations)**

Est une méthode d'explicabilité des modèles de machine learning basée sur la théorie des jeux. Elle attribue à chaque caractéristique d'entrée une valeur appelée valeur de Shapley qui mesure la contribution de cette caractéristique à la prédiction finale du modèle.

Cette méthode se distingue par sa neutralité vis-à-vis du modèle utilisé, ce qui signifie qu'elle peut s'appliquer à n'importe quel modèle, qu'il soit simple ou complexe. SHAP calcule ces valeurs en décomposant la prédiction comme la somme des impacts individuels des caractéristiques, permettant ainsi une compréhension fine et cohérente des comportements du modèle même en présence d'interactions complexes entre caractéristiques.

L'un des grands avantages de SHAP est d'expliquer les prédictions de façon compréhensible pour un humain, en montrant comment chaque variable influence positivement ou négativement la sortie, ce qui facilite l'interprétation, la détection de biais, et la validation des modèles en contexte réel.[13]

**Résultats et interprétation :**

**3.4.3 LIM: Local Interpretable Model-agnostic Explanations**

Est une technique d'interprétabilité qui approximativement simplifie localement un modèle complexe autour d'une prédiction donnée pour en expliquer le fonctionnement. Plutôt que d'analyser le modèle dans sa globalité, LIME se concentre sur une instance spécifique et génère des données artificielles proches de cette instance à travers de petites perturbations. Ces instances modifiées sont utilisées pour entraîner un modèle interprétable simple, souvent une régression linéaire, qui permet de comprendre quelles caractéristiques influencent la prédiction.

LIME repose sur quatre principes majeurs : facilité de compréhension, fiabilité (minimiser l'écart d'explication avec le modèle original), flexibilité (fonctionne avec tout type de modèle) et généralisation (expliquer aussi bien une prédiction unique qu’un sous-ensemble via SP-LIME).

Cette méthode est avantageuse pour expliquer des prédictions locales notamment dans des contextes où la transparence est cruciale.[14]

**Résultats et interprétation :**

**Conclusion :**

**Bibliographie :**

[1]: <https://www.kaggle.com/datasets/imdevskp/corona-virus-report?select=covid_19_clean_complete.csv>

[2]: https://www-sop.inria.fr/members/Sylvain.Chevillard/fr.selfhtml.org/graphisme/techniques.htm

[3]: https://fr.wikipedia.org/wiki/Contraste

[4]: https://www.ibm.com/fr-fr/think/topics/downsampling

[5]: https://www.ibm.com/fr-fr/think/topics/upsampling

[6]: https://www.lakera.ai/ml-glossary/baseline-models

[7]: <https://blent.ai/blog/a/modele-machine-learning-populaires>

[8]: <https://datascientest.com/fine-tuning-tout-savoir>

[9]: <https://fr.mathworks.com/help/deeplearning/ref/inceptionv3.html>

[10]: <https://fr.statisticseasily.com/glossaire/qu%27est-ce-que-la-recherche-par-grille/>

[11]: <https://blent.ai/blog/a/xgboost-tout-comprendre>

[12]: <https://datascientest.com/en/what-is-the-grad-cam-method>

[13]: <https://datascientest.com/shap-tout-savoir>

[14]: <https://interpret.ml/docs/lime.html>

# **Annexe :**

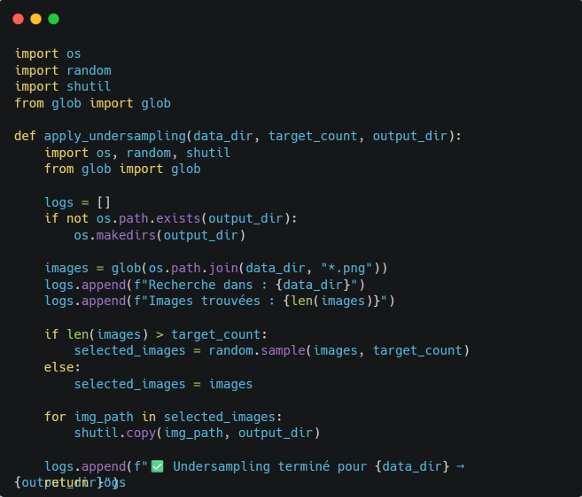
# **Annexe 1 : Sélecteur de dossiers pour Jupyter**



**Figure 01 :** Code de select\_folders.py

Le module select\_folders.py implémente la classe FolderSelector, un widget interactif pour Jupyter Notebook permettant de sélectionner des dossiers. Conçu avec ipywidgets et IPython.display, il remplace l’usage de input() par une interface intégrée : champ texte pour saisir un chemin, validation en temps réel (existence du dossier), affichage des chemins ajoutés et feedback visuel. Un bouton Terminer valide la sélection, cache l’interface et laisse les chemins accessibles via l’attribut selector.chemins lorsque selector.fini == True. Le widget améliore l’ergonomie du notebook en fournissant une expérience utilisateur fluide, un retour immédiat sur les saisies et une gestion automatique de la fermeture de l’interface.

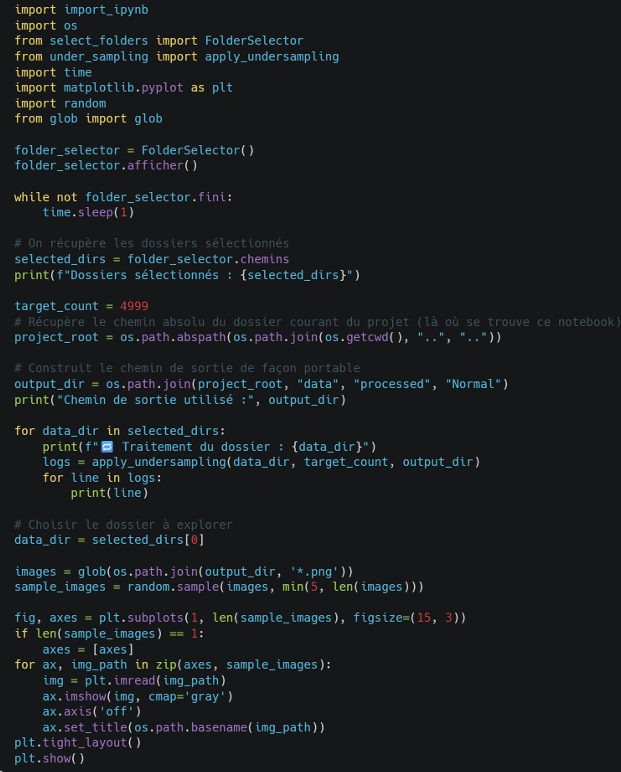
# **Annexe 2 : Undersampling d’images**



**Figure 02 :** Code de under\_sampling.ipynb

Le notebook under\_sampling.ipynb contient une fonction apply\_undersampling(data\_dir, target\_count, output\_dir) qui réduit aléatoirement le nombre d’images d’un dossier source afin d’obtenir target\_count images en sortie. La fonction liste les fichiers .png du dossier source, sélectionne aléatoirement — si nécessaire — target\_count images, crée le dossier de sortie s’il n’existe pas, puis copie les images sélectionnées. Elle renvoie une liste de logs décrivant les opérations effectuées. Usages typiques : rééquilibrage d’un jeu de données, réduction de la taille d’un dataset trop volumineux et préparation de données pour un pipeline d’apprentissage automatique.

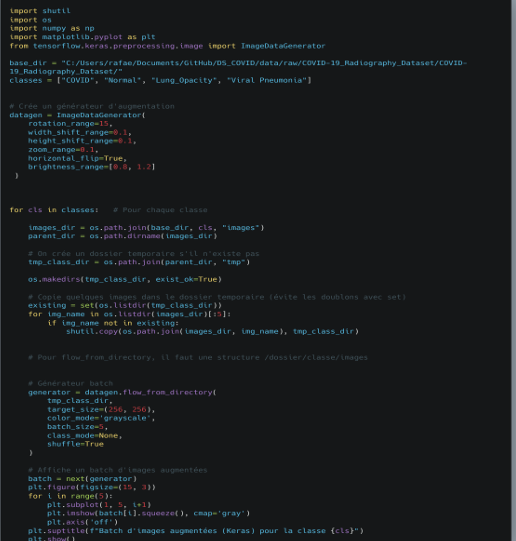
# **Annexe 3 : Pipeline principal**



**Figure 03 :** Code de Main.ipynb

Le notebook Main.ipynb implémente le script principal orchestrant le prétraitement d’images : sélection interactive des dossiers via FolderSelector, application d’un undersampling (apply\_undersampling) pour réduire chaque jeu à target\_count = 4999 images, copie des images sélectionnées dans un dossier de sortie commun (data/processed/Normal) construit de façon portable, puis validation visuelle en affichant cinq images aléatoires du dataset final avec matplotlib. Ce pipeline vise à automatiser la préparation de datasets équilibrés dans l’environnement Jupyter tout en offrant une interface utilisateur intuitive et des mécanismes de vérification visuelle.

# **Annexe 4 : Oversampling avec Keras**

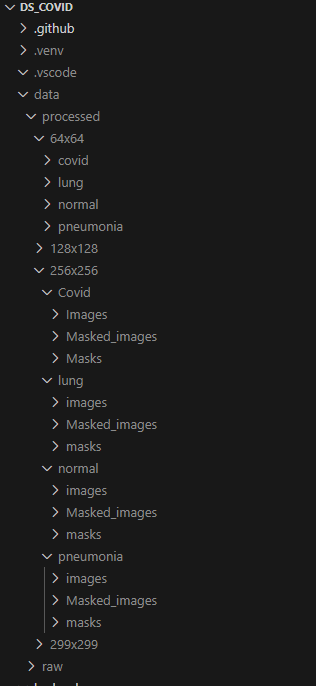


**Figure 04 :** Codede over\_sampling.ipynb

Le notebook over\_sampling.ipynb expérimente l’oversampling via Keras en utilisant ImageDataGenerator et sa méthode flow\_from\_directory. Plutôt que de dupliquer manuellement des fichiers, ImageDataGenerator permet de produire des images augmentées à la volée pendant l’entraînement : chargement par batch, redimensionnement automatique (target\_size), conversion du mode couleur (color\_mode) et application d’opérations d’augmentation (rotation, flip, zoom, etc.). Les valeurs utilisées dans le notebook sont issues de la documentation Keras et restent paramétrables. L’approche facilite la connexion directe avec la phase de modélisation (fit/generator) et accélère le prototypage d’un workflow complet d’entraînement.

**Annexe 05 : Arborescence Automatique**

Afin de pouvoir garder un environnement propre, une arborescence automatique pour trier les éléments générés a été mise en place.  
Celle-ci génère dans le dossier ”processed”: Un premier dossier en fonctions de la résolution,  
Exemple : 256x256 (Nom du dossier).  
Ensuite, dans ce dossier, 4 autres sont créées pour chacune (covid,lung,normal,pneumonia)  
Dans chacun de ces dossiers, on va générer par type d’éléments générés :  
 - Images redimensionnés   
 - Masques redimensionnés  
 - Images avec Masque appliqué



**Figure 05 :** Exemple de l’Arborescence Automatique