



Classification de Tumeurs Cérébrales à l'aide du ML/DL

Réalisé et présenté par :

- MESDOUR Yasmine.
- KOUHIL Zakaria.
- ZEMMOURI Yasmine.
- GHAZOUI Ilyas.

Encadré par :

Feda Almuhsen

Sommaire :

Présentation du
projet.

Méthodologie et
pipeline.

Jeu de données.

01

02

03

04

05

06

Classification de
Tumeurs Cérébrales à
l'aide du
Machine Learning et
DeepLearning

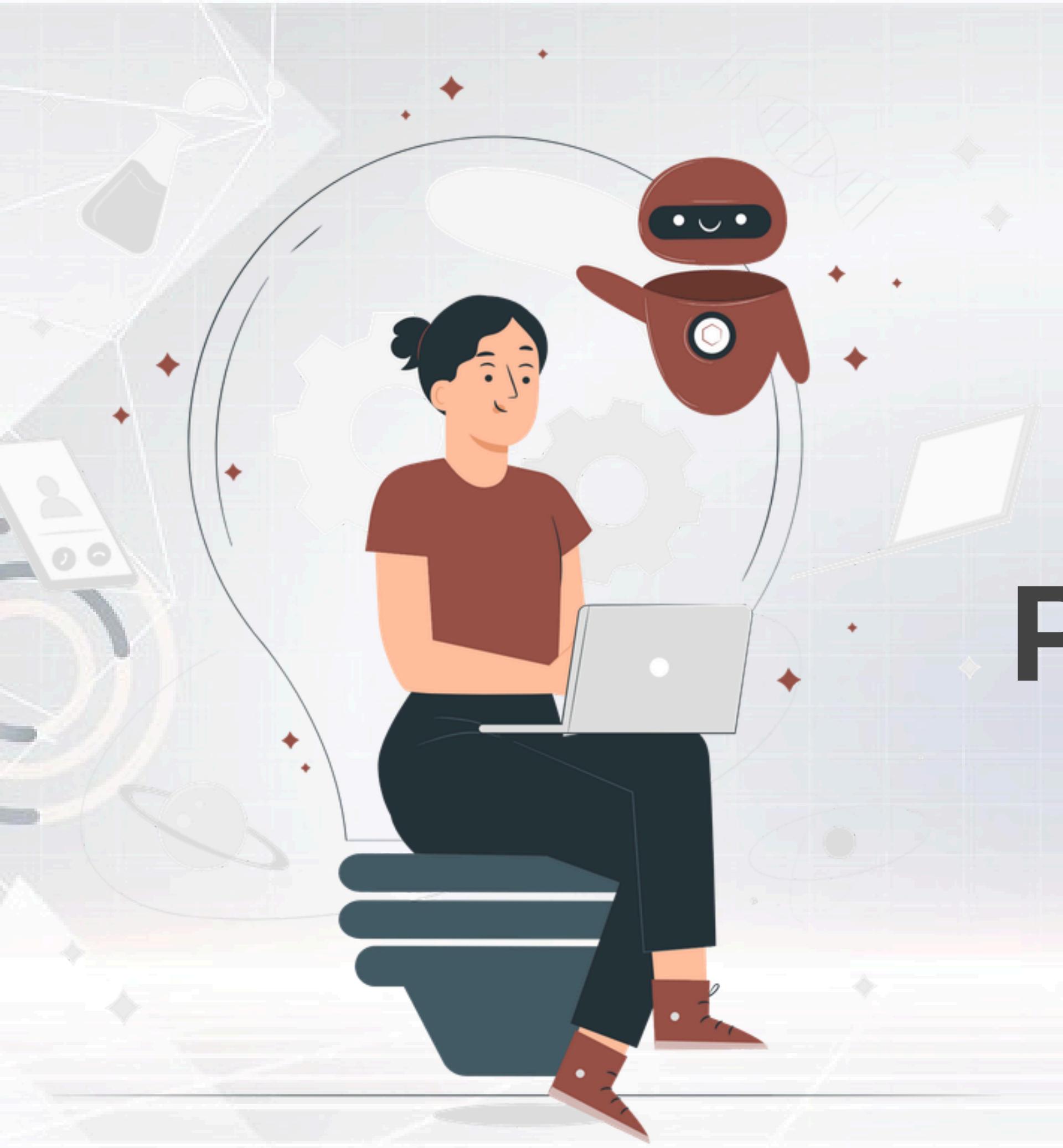
Modèles utilisés.

Résultats.

Conclusion.

01

Présentation du projet



Présentation

**Machine
Learning**



Quelle tumeur est présente ?

Comment la classer ?

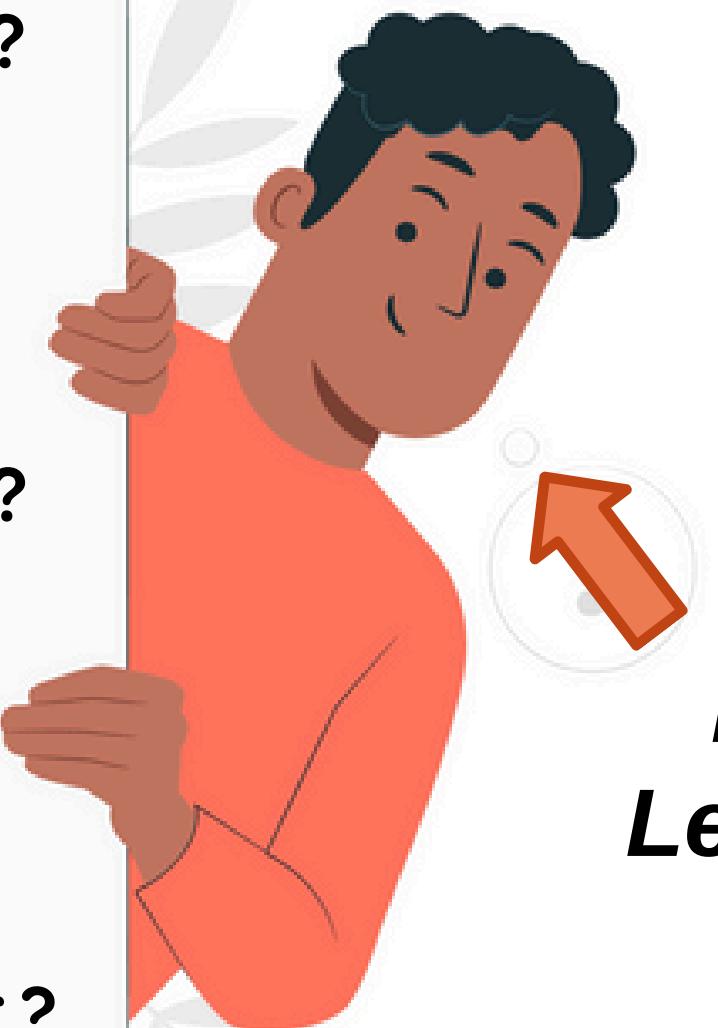
A partir d'images ?

IA pour la médecine ?

ML ou DL ?

Qui est le meilleur ?

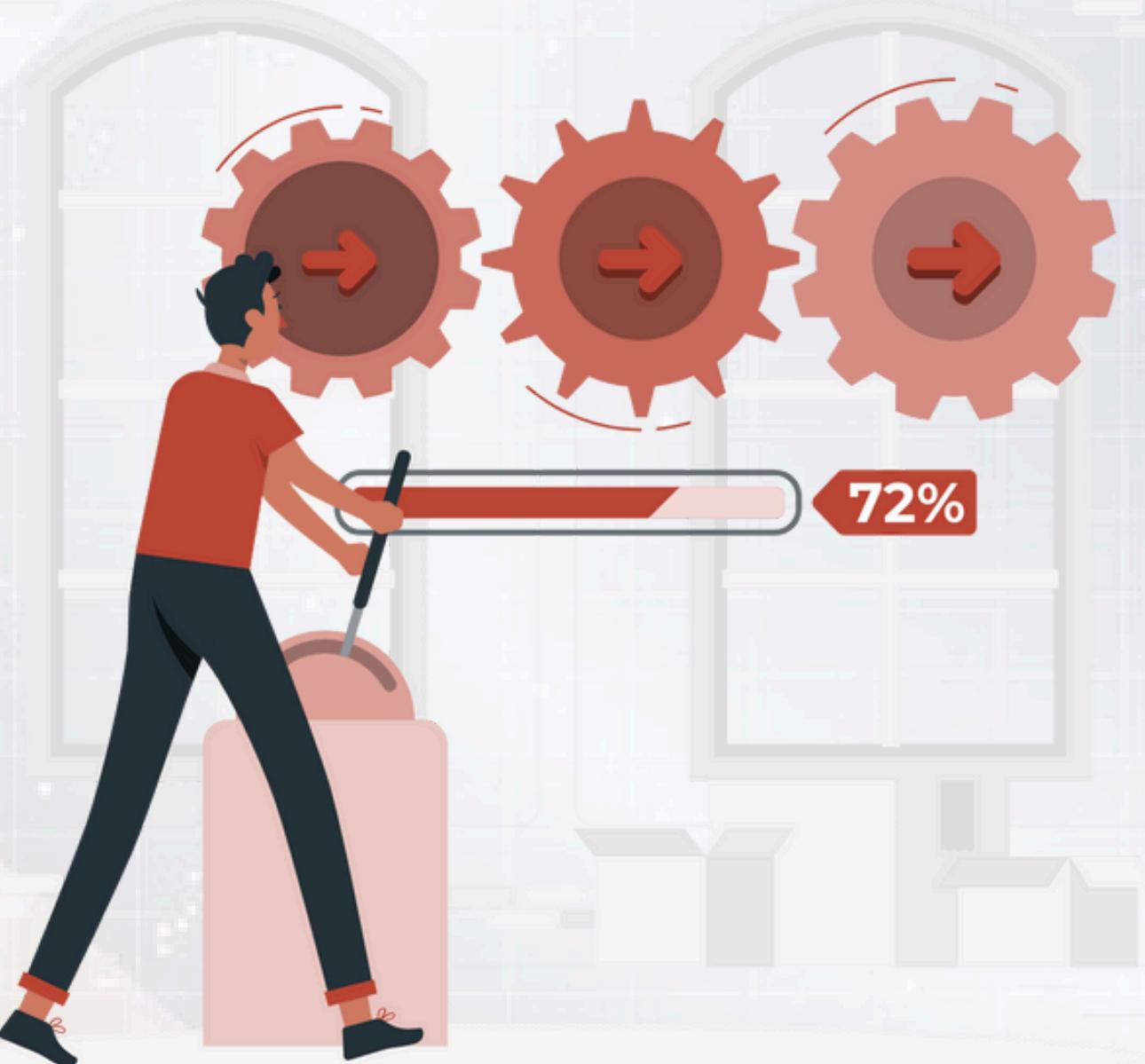
Une multi-classification ?



**Deep
Learning**

02

Méthodologie et pipeline



Méthodologie et Pipeline

01

Préparation des données :

- Prétraitement des données.
- Gestion des déséquilibres.

02

Entraînement des modèles :

- Random Forest, CNN, et Shallow Network.
- Optimisation des hyperparamètres via une Validation Croisée (3-fold).
- **Suivi des expériences avec MLflow :**
- Chaque exécution du modèle est enregistrée (hyperparamètres, métriques).
- Suivi des métriques pour chaque pli de validation croisée (Accuracy, Precision, Recall, F1-Score).
- Sauvegarde et gestion des versions du modèle final.

03

Évaluation des modèles :

- Utilisation de métriques adaptées : Accuracy, Precision, Recall, F1-Score, AUC.
- **Analyse de l'interprétabilité avec LIME :**
- LIME : Analyse locale pour comprendre les contributions des caractéristiques dans des cas spécifiques

04

Comparaison et sélection :

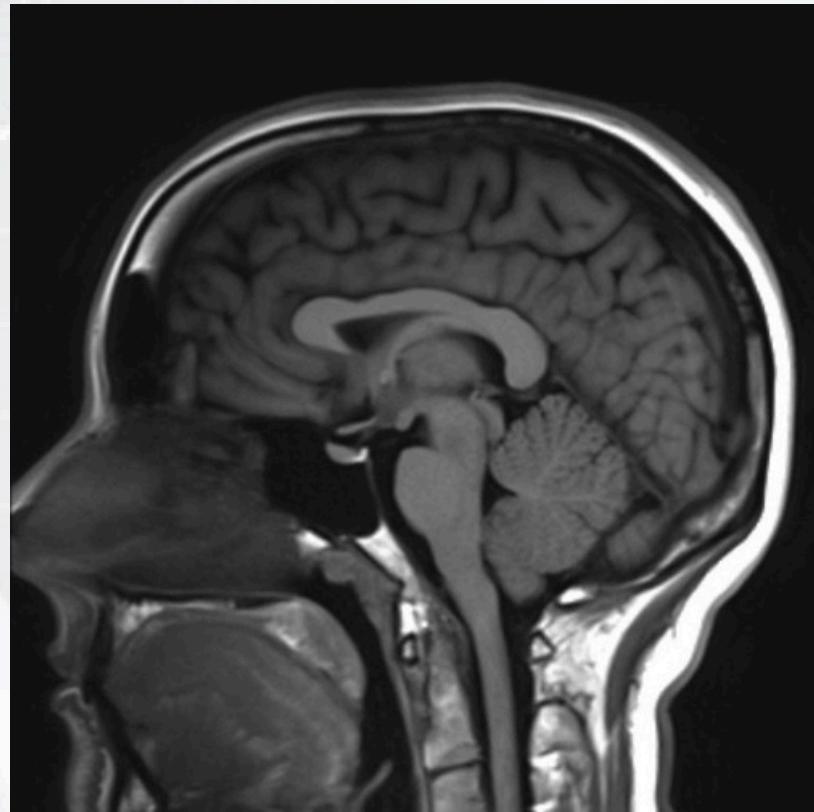
- Analyse comparative des performances des modèles.

03 Jeu de données

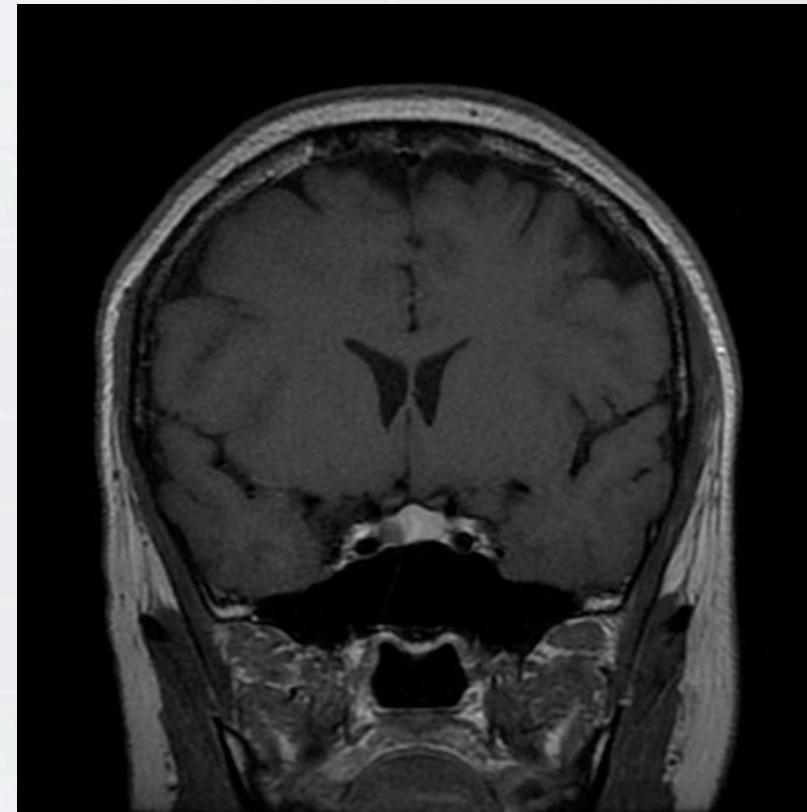


Brain Tumor MRI Dataset

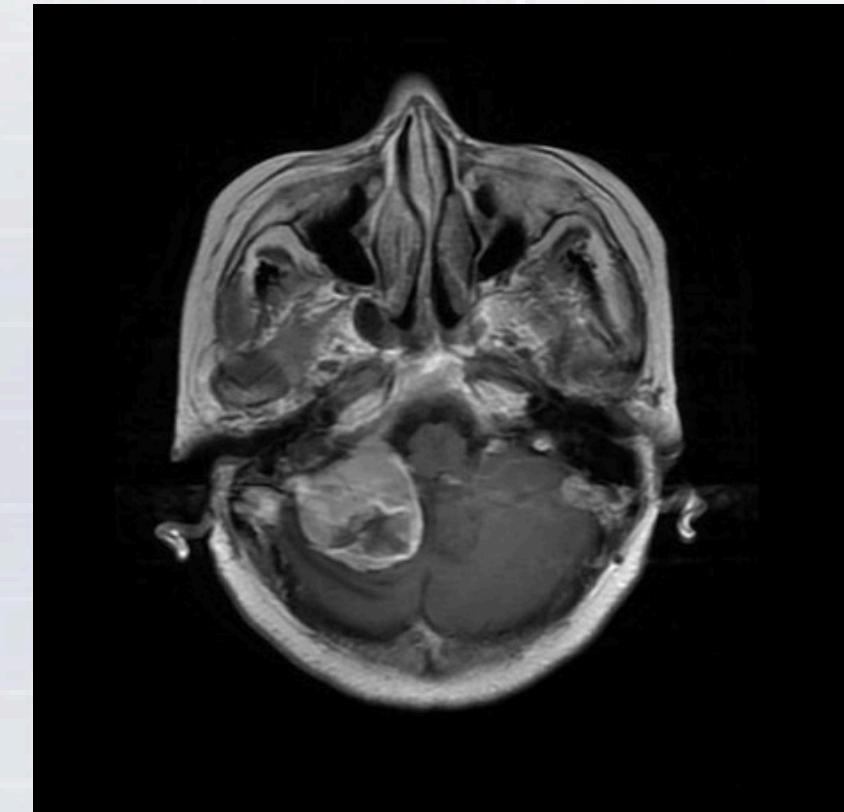
Nous avons utilisé le **Brain Tumor MRI Dataset** de **Kaggle** avec **7023** images réparties en 4 classes :
notumor - pituitary - meningioma - glioma



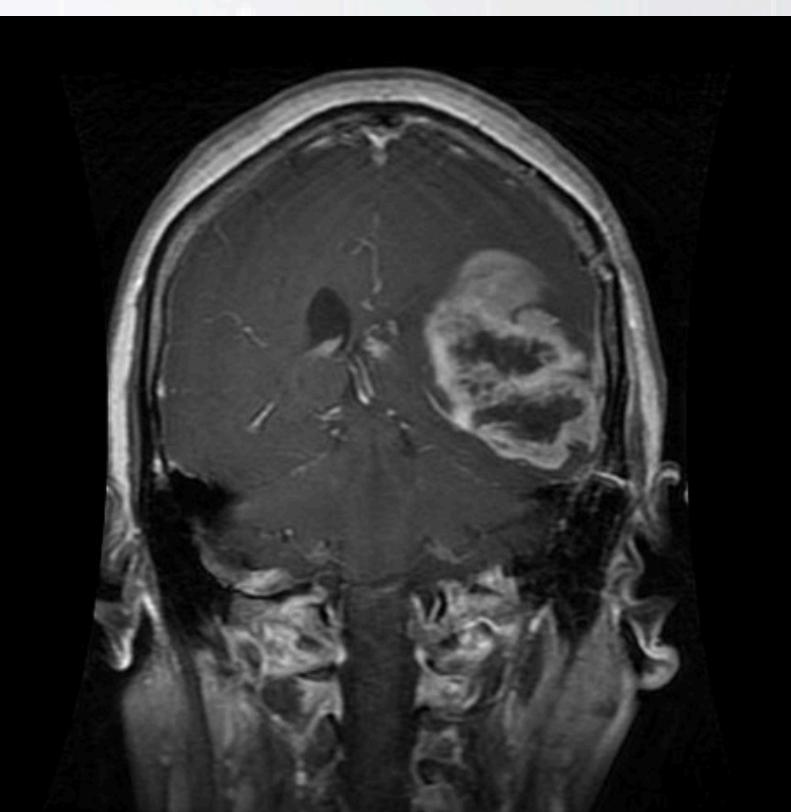
notumor



pituitary



meningioma



glioma

Prétraitement

01

Pas de données manquantes : les données sont des images (matrice de pixels).

02

Resize les images en (128,128) : nos données n'ont pas une taille unique :
 $\{(491, 624), (206, 244), (409, 442), (503, 369), (430, 483), \dots\}$.

03

Encodage des classes: nos classes sont de type texte, un encodage simple en valeur numérique a été fait :

- 'notumor': 0,
- 'pituitary': 1,
- 'meningioma': 2,
- 'glioma': 3.

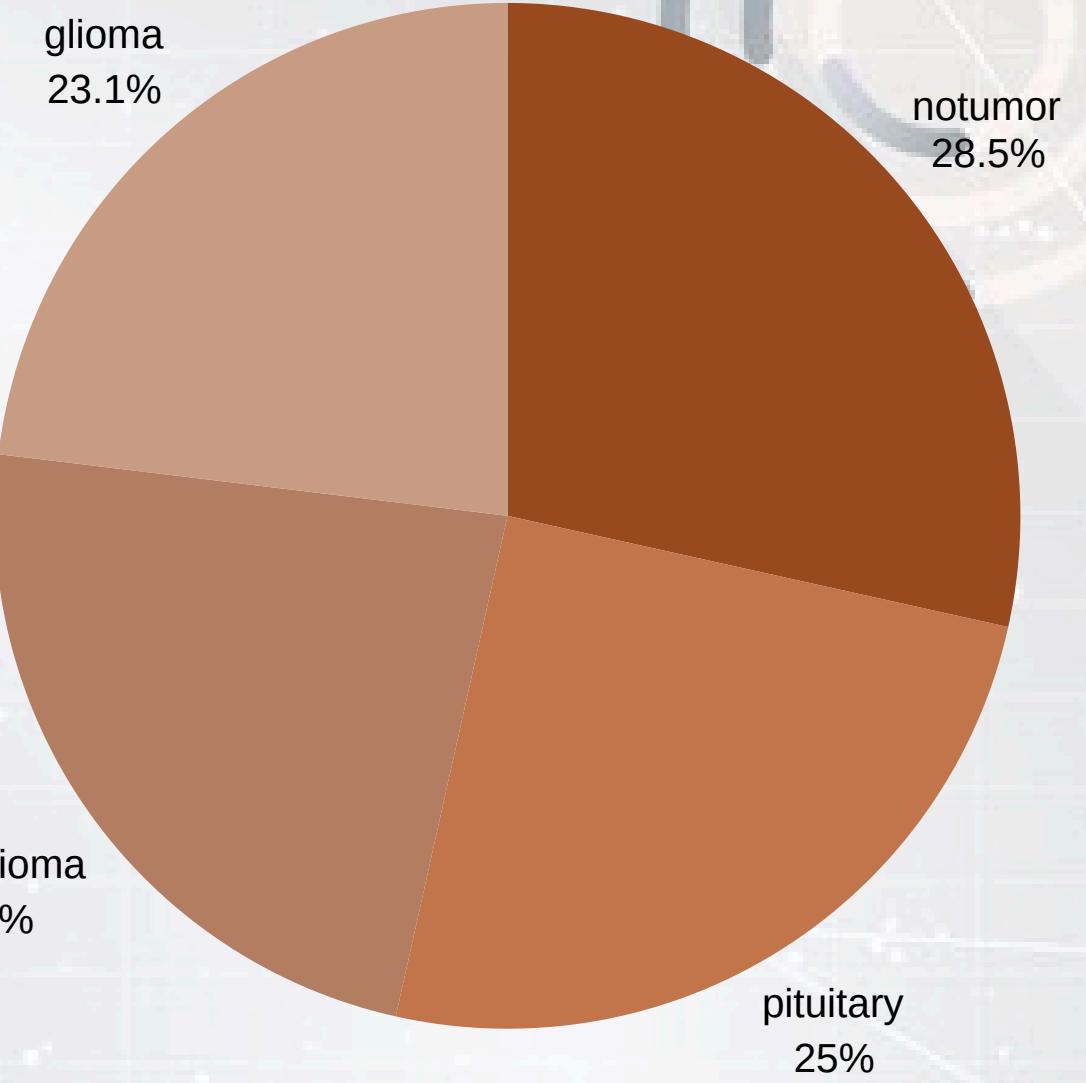
Prétraitement

04

Rééquilibrage des données : Le dataset est légèrement déséquilibré :

- notumor - 2000 images.
- pituitary - 1757 images.
- meningioma - 1645 images.
- glioma - 1621 images.

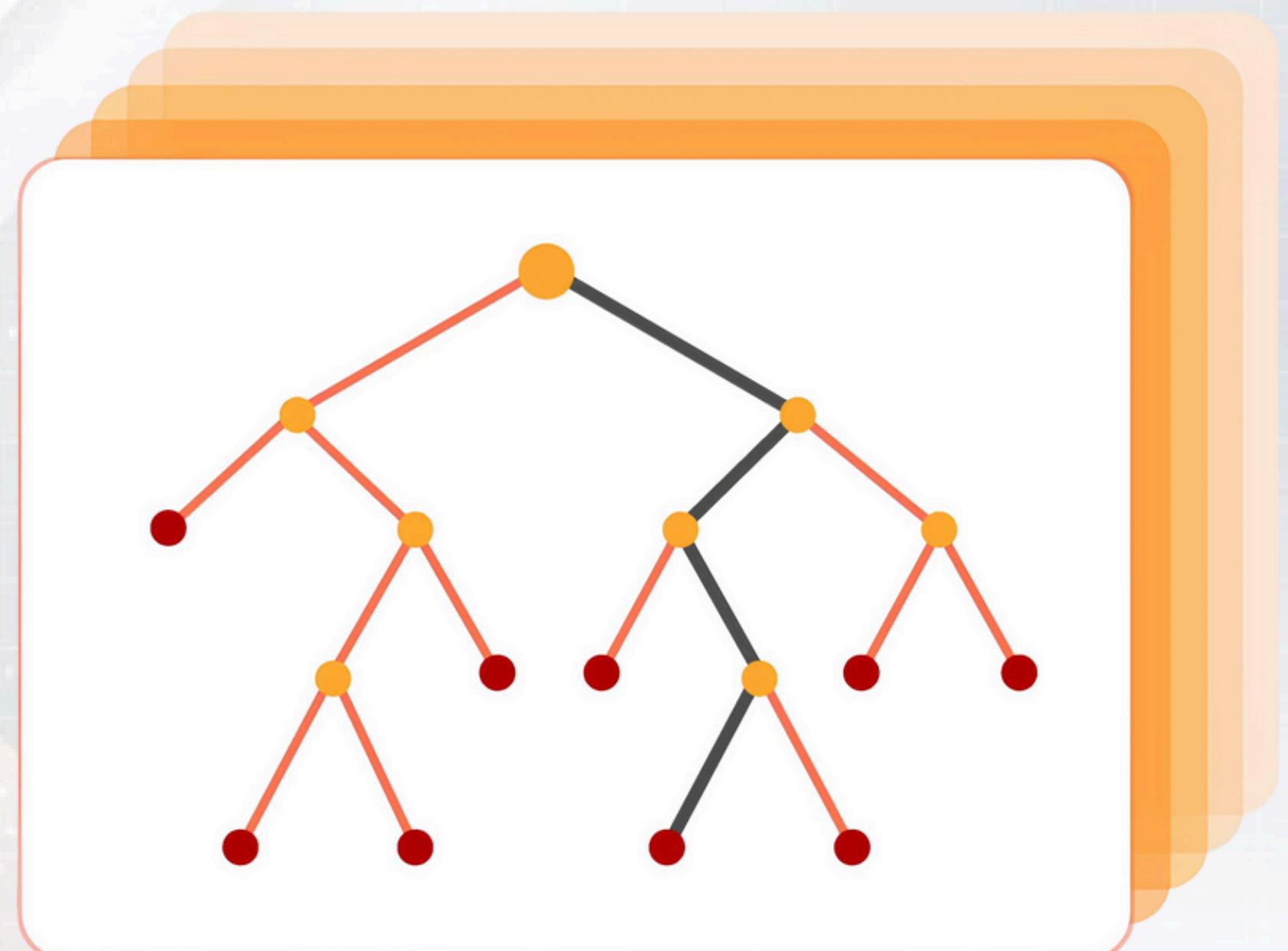
Nous avons appliquer un **sur-échantillonnage** sur nos données pour atteindre **2000 échantillons par classe**, en appliquant une rotation à 90° , 180° ou 270° sur quelques une de nos images déjà existantes.



04 Modèles utilisés

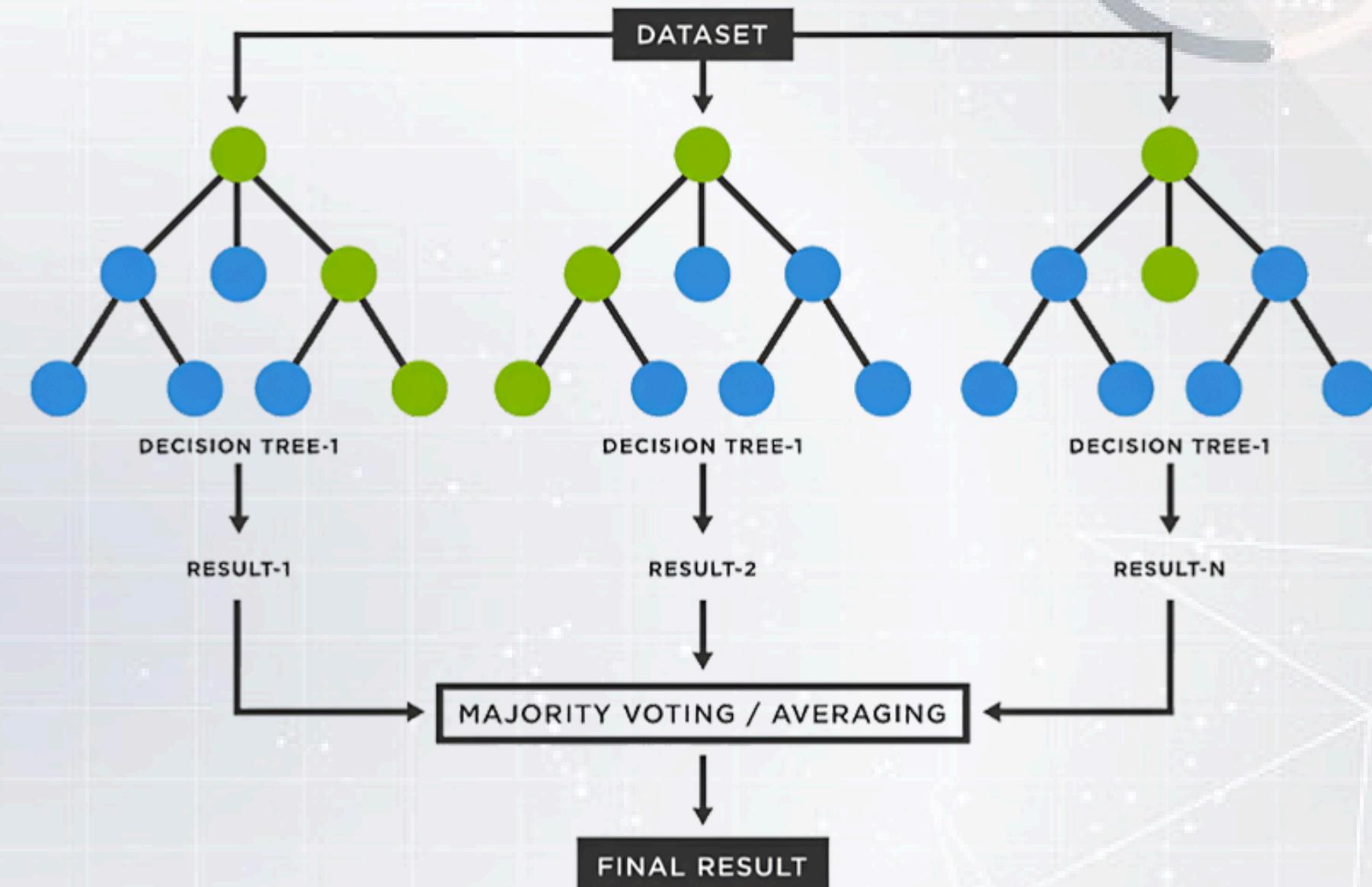


Random Forest



Random Forest

La Random Forest est un algorithme d'apprentissage automatique qui combine plusieurs arbres de décision pour faire des prédictions. Chaque arbre est construit sur un échantillon aléatoire des données et utilise une sélection aléatoire des caractéristiques.



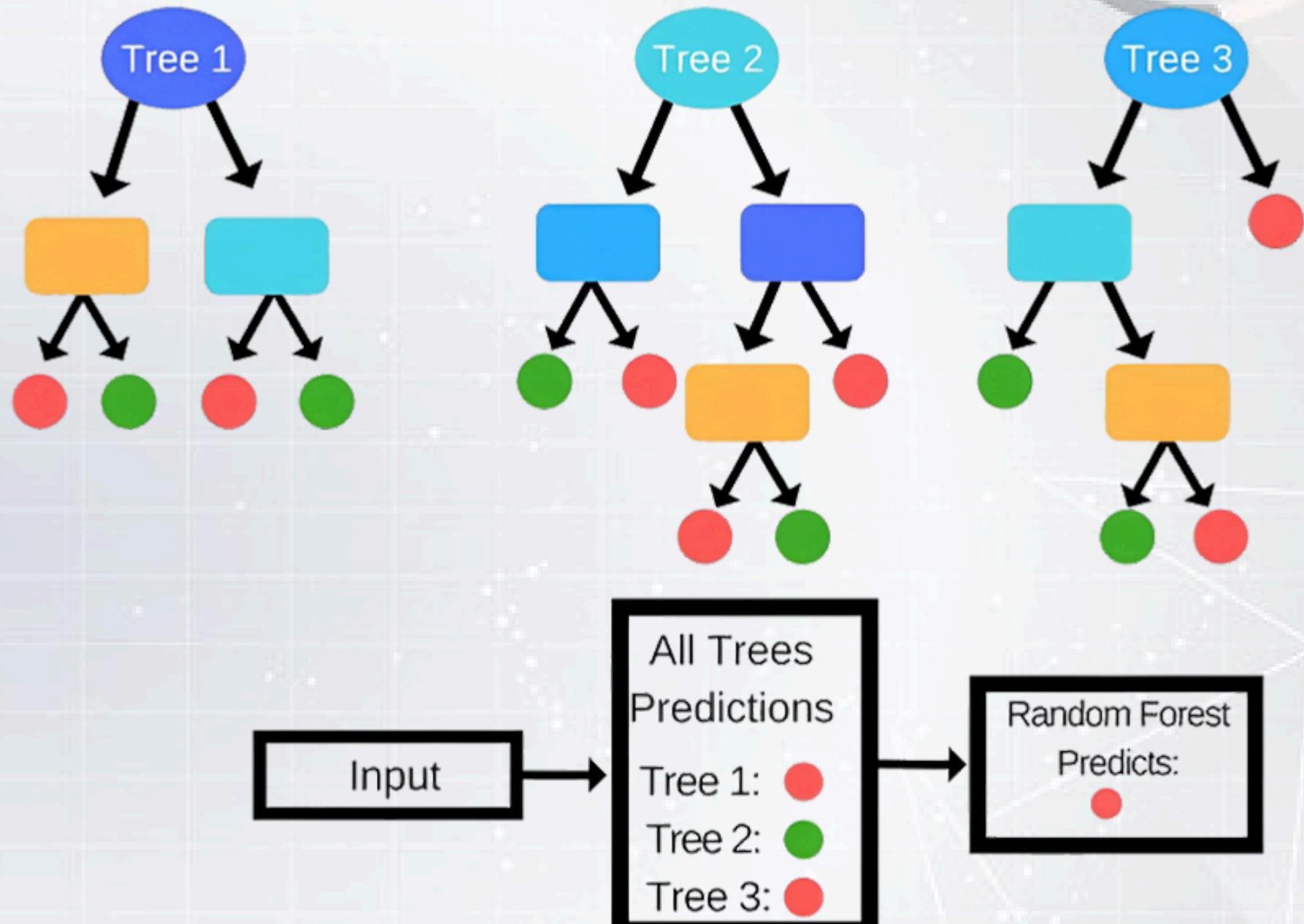
Random Forest

Pour la classification, le vote majoritaire dans une Random Forest est déterminé en comptant le nombre de votes pour chaque classe parmi tous les arbres, puis en choisissant la classe ayant le plus de votes.

Exemple simple :

- 2 arbres votent pour la classe Rouge.
- 1 arbre vote pour la classe Vert.

Le modèle prédit la classe Rouge, car elle a obtenu le plus de votes ($2 > 1$).



Pourquoi Random Forest ?

Avantages de la Random Forest pour la Classification des Tumeurs Cérébrales

- Gère efficacement les données à haute dimension.
- Robuste face aux anomalies.
- Identifie les caractéristiques importantes.
- Convient aux petits ensembles de données.
- Réduit le surapprentissage.
- Pas besoin de prétraitement complexe.

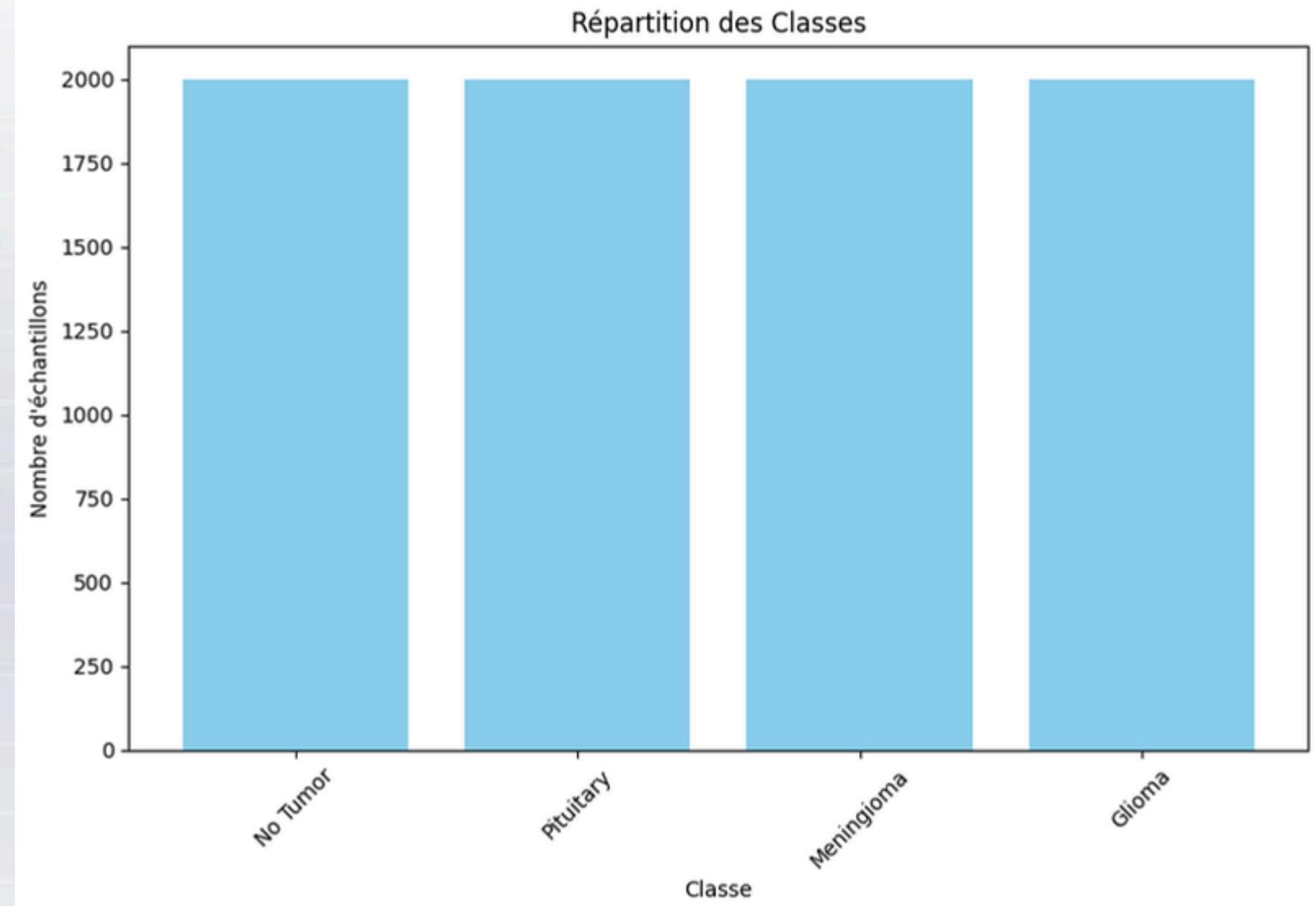


Dataset

Vérification de la Distribution des Données :

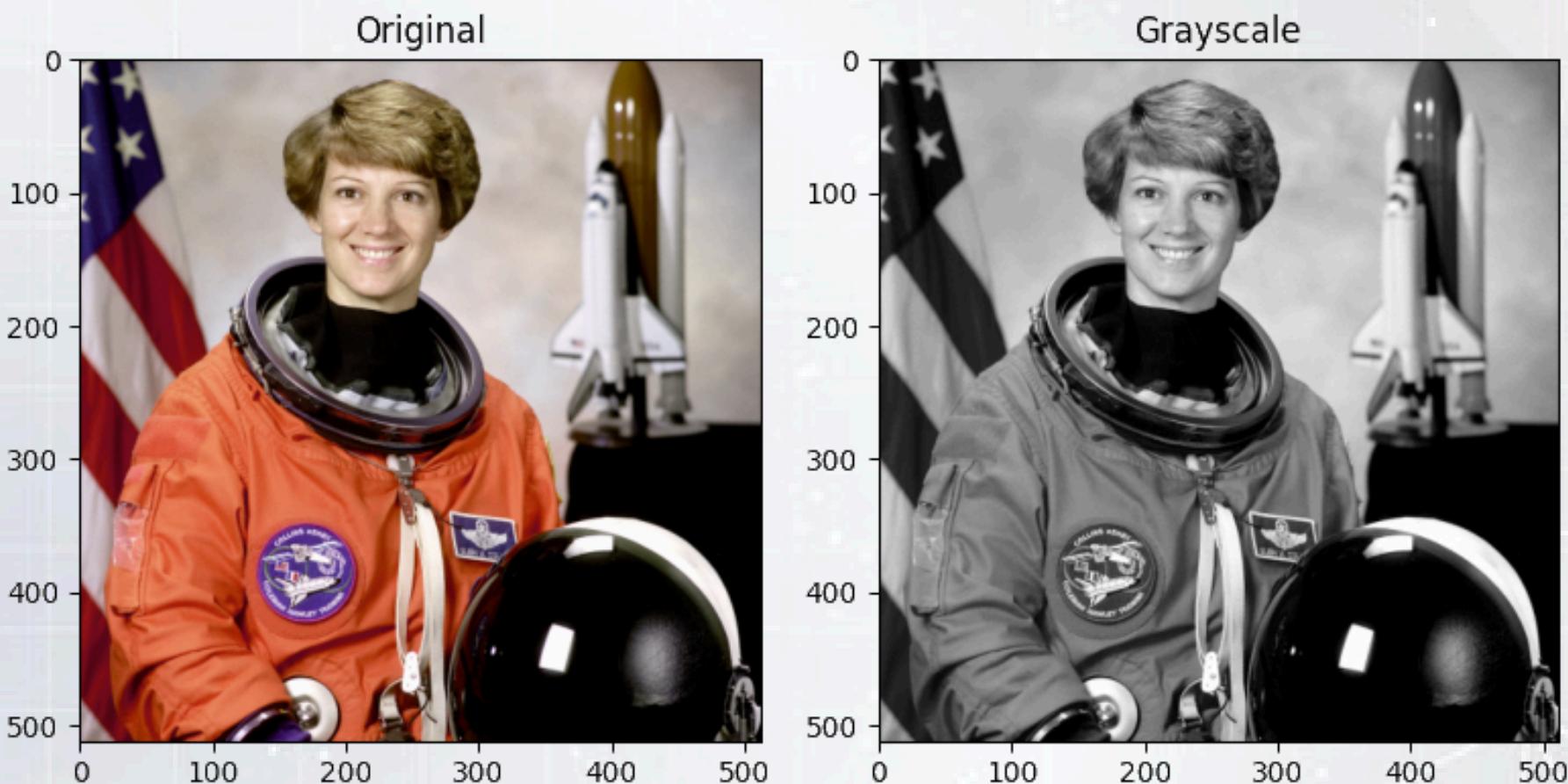
- *Objectif* : Éviter le déséquilibre entre les classes.
- Vérification effectuée après le prétraitement.
- *Résultat* :
 - Chaque classe contient 2000 images.
 - Dataset total : 8000 images.

Donc dataset équilibré et suffisant pour entraîner le modèle.



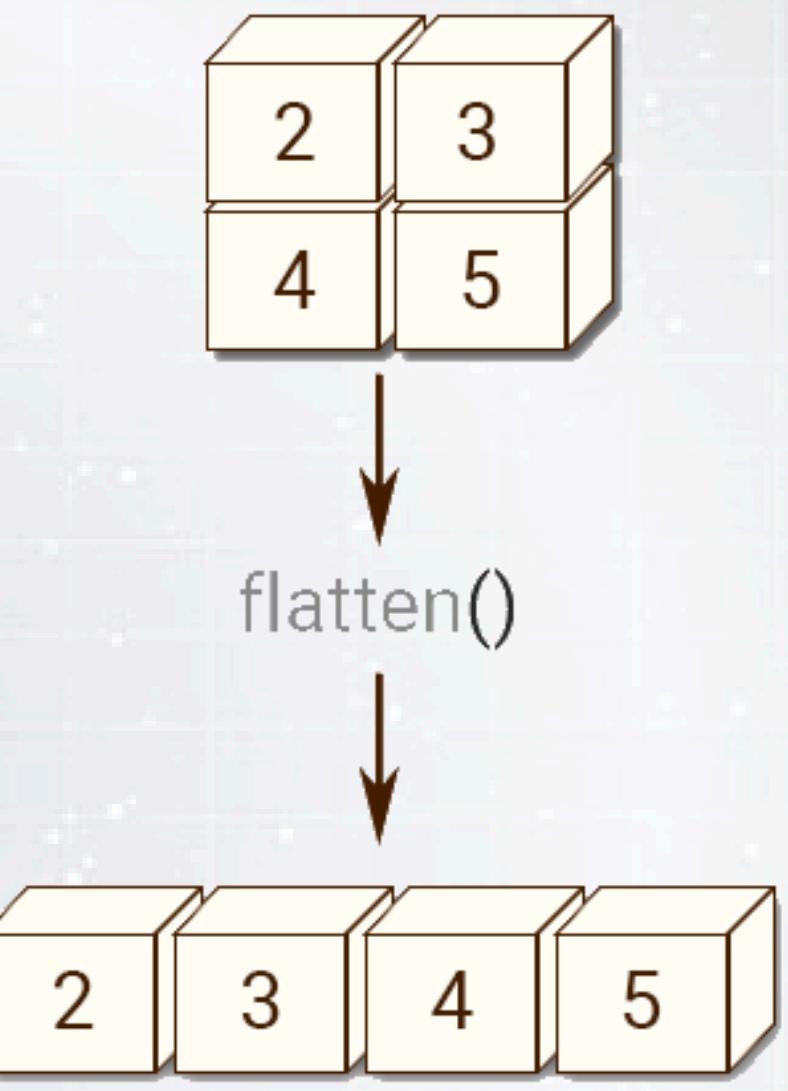
RGB to grayscale

Pour simplifier les données, nous avons converti les images en niveaux de gris en utilisant la fonction `rgb2gray()`. Cela réduit les trois canaux de couleur (rouge, vert, bleu) en une seule dimension, en conservant les informations nécessaires à l'analyse et à la classification.



2D image to 1D

Après avoir converti les images en niveaux de gris avec la fonction **`rgb2gray()`**, nous les transformons en vecteurs à l'aide de la fonction **`flatten()`**. Cela permet de convertir l'image **2D** en une structure linéaire **1D**, facilitant ainsi son utilisation comme entrée pour notre modèle d'apprentissage automatique.



Entrainement du modèle

Nous lançons l'entraînement de notre modèle Random Forest en utilisant les paramètres par défaut

- *n_estimators = 100*
- *max_depth = None*
- *min_samples_split = 2*
- *max_features = sqrt*

Ce qui est souvent suffisant pour obtenir de bonnes performances sans trop de complexité.

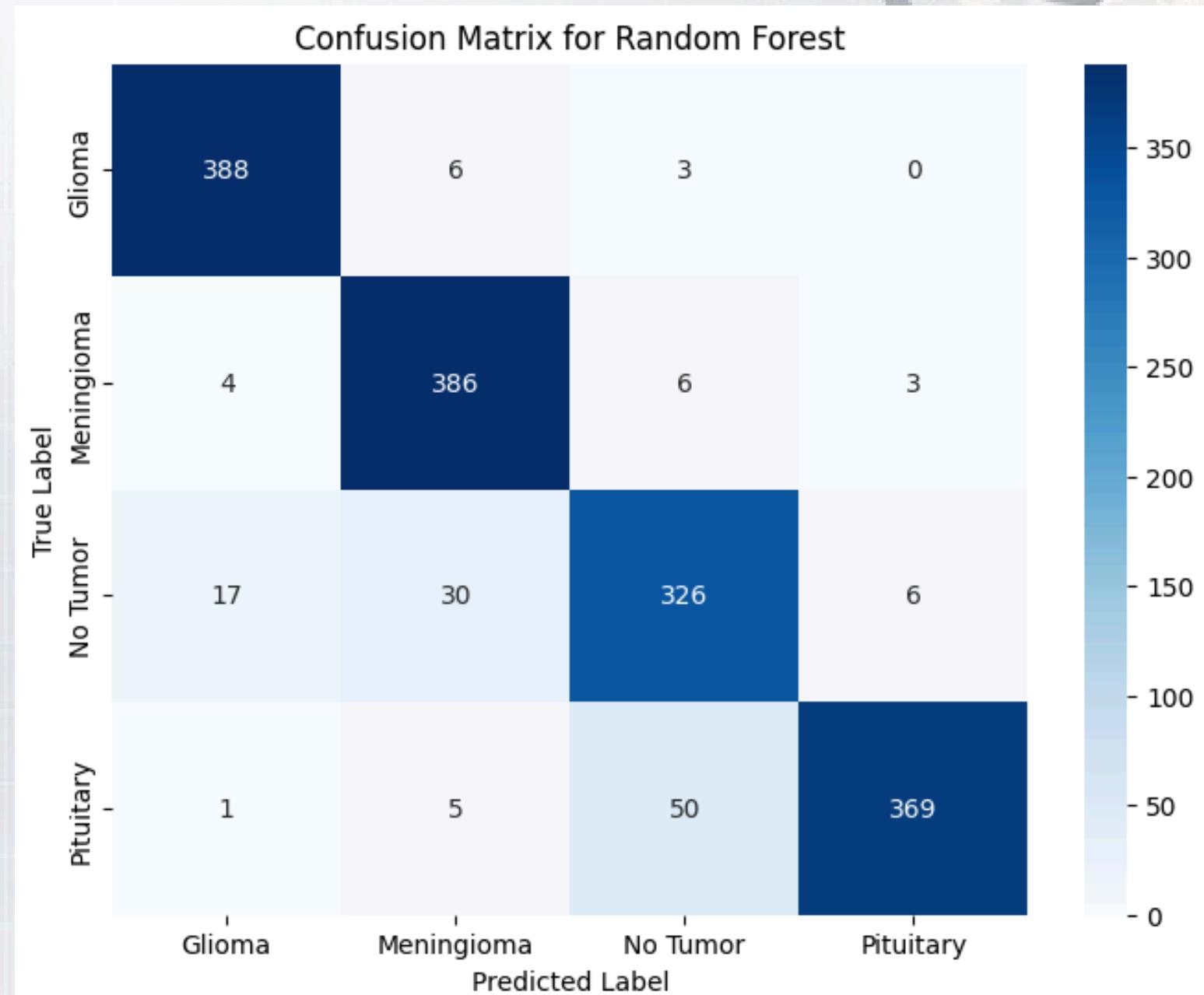
```
# Define the Random Forest model
model = RandomForestClassifier()

# Train the model on the training data
model.fit(X_train, y_train)

# Evaluate the model on the test data and print the accuracy
accuracy = model.score(X_test, y_test)
print(f"Accuracy of Random Forest: {accuracy}")
```

Résultat du modèle

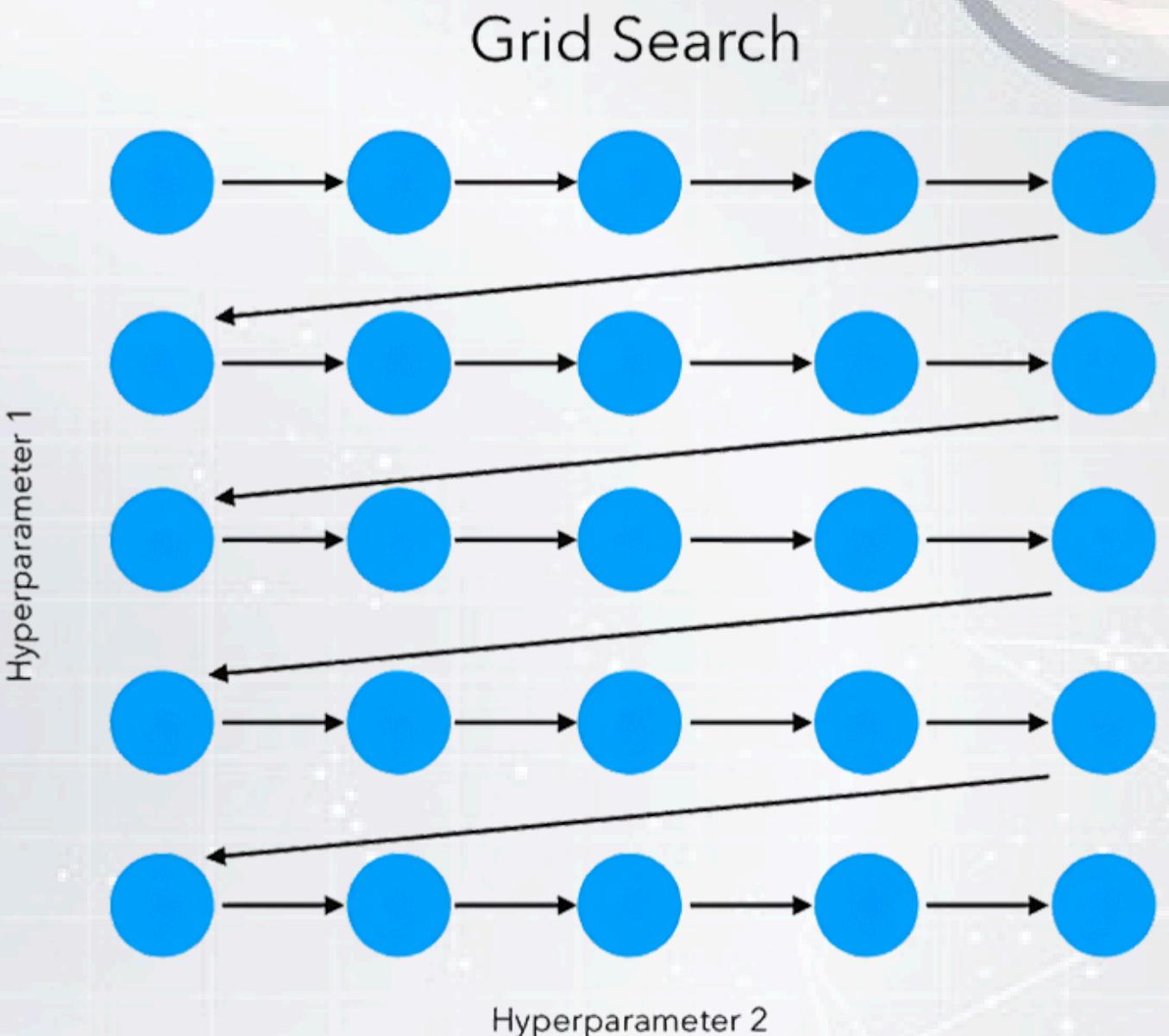
*Nous avons obtenu une accuracy de **0.918** avec les paramètres par défaut du Random Forest. La matrice de confusion montre cependant que le modèle peut encore être amélioré pour de meilleurs résultats.*



Améliorer le modèle

Pour améliorer le modèle, nous avons utilisé le même concept du **Grid Search** afin d'explorer plusieurs combinaisons de paramètres et trouver les meilleurs paramètres adaptés à notre modèle et à notre dataset avec les paramètres suivants :

- `n_estimators` : de 10 à 201 (pas de 20)
- `max_depth` : [None, 10, 20, 30]
- `min_samples_split` : [2, 5, 10]
- `max_features` : ['sqrt', 'log2']



Améliorer le modèle

Après l'entraînement et l'évaluation de chaque configuration, les meilleurs résultats ont été obtenus avec les paramètres suivants :

- *n_estimators* : 30
- *max_depth* : None
- *min_samples_split* : 2
- *max_features* : *sqrt*

Brain_Tumor_Classification_Random_Forest

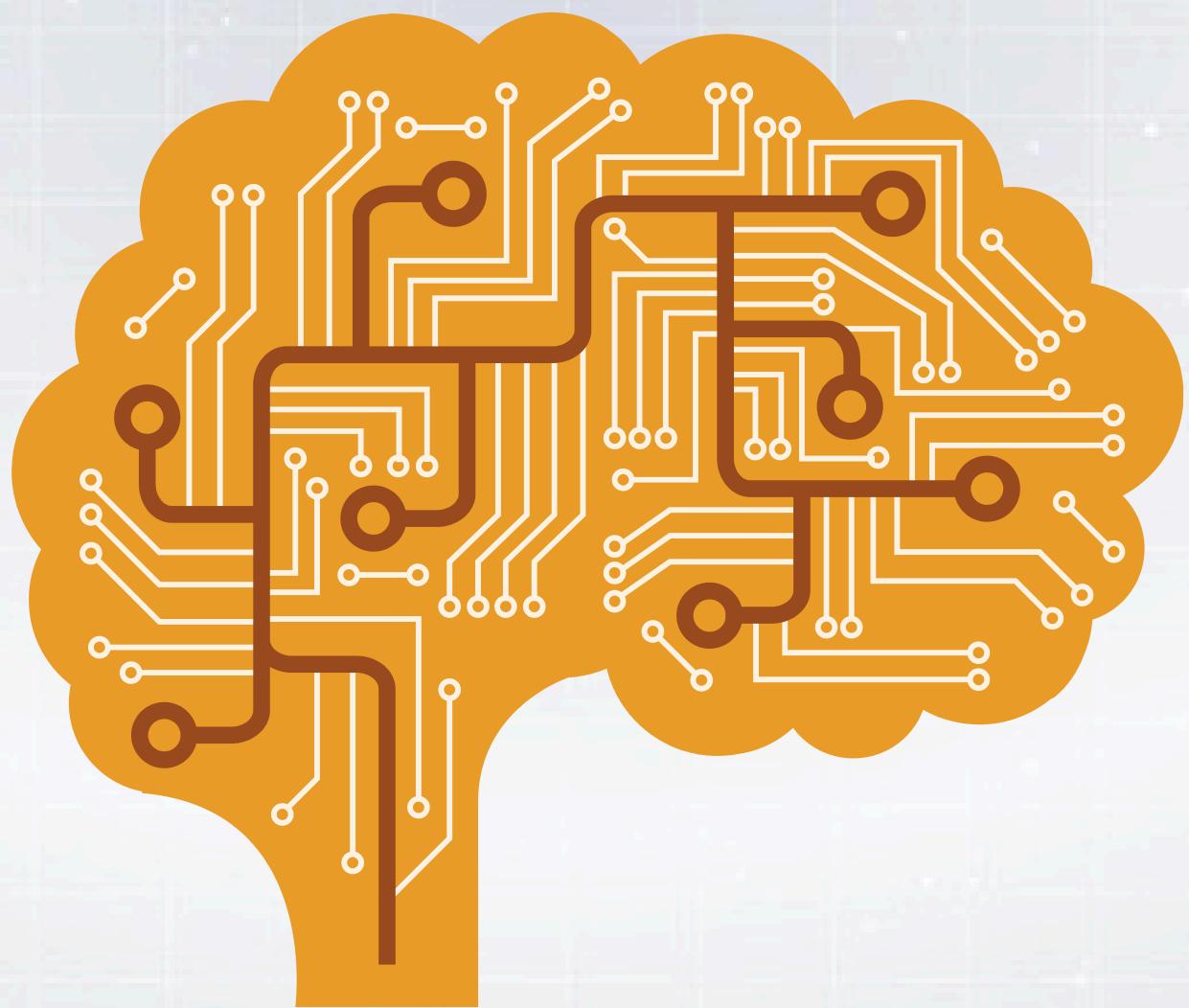
Run Name	Created	Dataset	Duration	Source	Models	accuracy	f1_score	precision	recall
colorful-seal-169	7 hours ago	-	19.9s	Cl/Cours...	-	0.919375	0.91915188...	0.92160843...	0.919375
stylish-squirrel-538	8 hours ago	-	17.9s	Cl/Cours...	-	0.919375	0.91915188...	0.92160843...	0.919375
worried-roo-715	10 hours ago	-	22.9s	Cl/Cours...	sklearn	0.919375	-	-	-
secretive-smelt-4	9 hours ago	-	1.1min	Cl/Cours...	sklearn	0.91875	-	-	-
thoughtful-sloth-902	9 hours ago	-	2.2min	Cl/Cours...	sklearn	0.91875	-	-	-
unruly-pig-545	9 hours ago	-	1.1min	Cl/Cours...	sklearn	0.91875	-	-	-
judicious-sow-461	9 hours ago	-	1.3min	Cl/Cours...	sklearn	0.918125	-	-	-
illustrious-croc-486	9 hours ago	-	53.3s	Cl/Cours...	sklearn	0.918125	-	-	-
clumsy-fish-982	9 hours ago	-	20.7s	Cl/Cours...	sklearn	0.918125	-	-	-
worried-fish-277	10 hours ago	-	1.2min	Cl/Cours...	sklearn	0.918125	-	-	-
welcoming-sloth-833	9 hours ago	-	1.5min	Cl/Cours...	sklearn	0.9175	-	-	-
intelligent-dog-515	9 hours ago	-	1.7min	Cl/Cours...	sklearn	0.9175	-	-	-
gentle-frog-102	9 hours ago	-	1.3min	Cl/Cours...	sklearn	0.9175	-	-	-
bouncy-horse-621	10 hours ago	-	1.4min	Cl/Cours...	sklearn	0.9175	-	-	-
merciful-stag-560	9 hours ago	-	1.8min	Cl/Cours...	sklearn	0.916875	-	-	-
merciful-gnu-717	9 hours ago	-	1.6min	Cl/Cours...	sklearn	0.916875	-	-	-
exultant-moth-145	10 hours ago	-	36.4s	Cl/Cours...	sklearn	0.916875	-	-	-
puzzled-lynx-234	8 hours ago	-	20.7s	Cl/Cours...	sklearn	0.91625	-	-	-
rogue-grouse-693	9 hours ago	-	42.7s	Cl/Cours...	sklearn	0.91625	-	-	-
valuable-cub-321	9 hours ago	-	31.5s	Cl/Cours...	sklearn	0.91625	-	-	-
youthful-flea-327	9 hours ago	-	12.3s	Cl/Cours...	sklearn	0.91625	-	-	-

Résultat du modèle

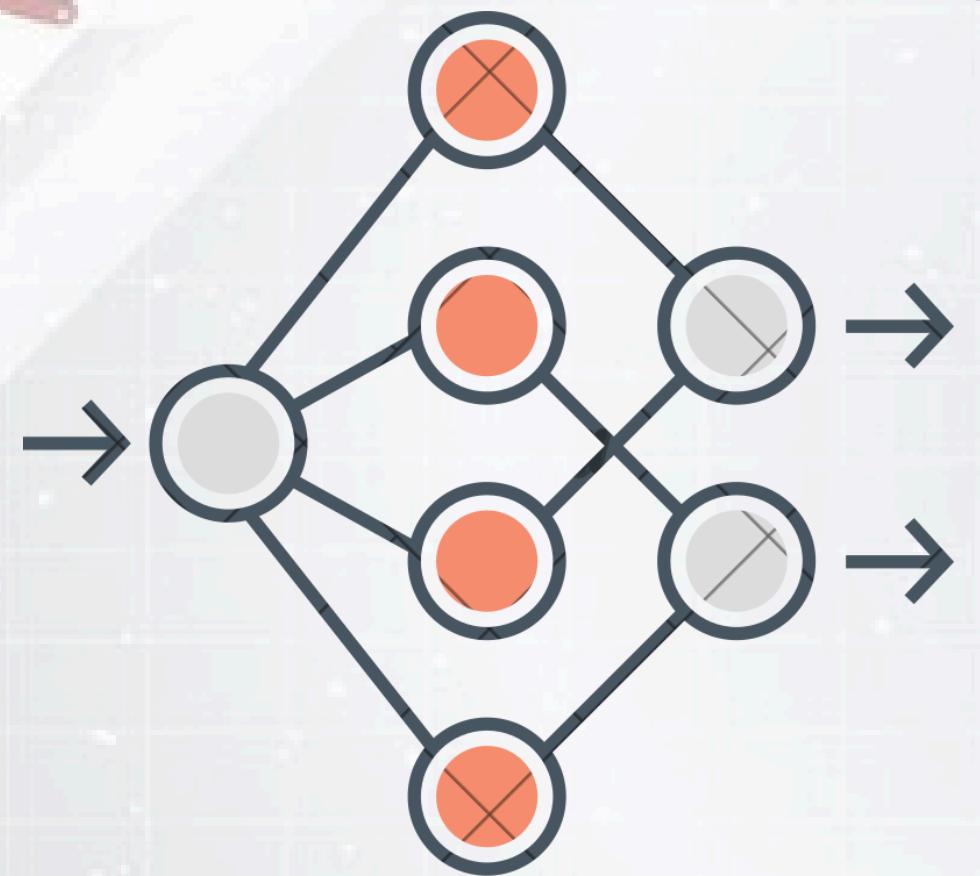
En utilisant cette configuration, nous avons entraîné notre modèle final et obtenu des performances optimales avec un taux de précision de **92%** sur nos données de test. Pour garantir une réutilisation et une traçabilité, le modèle a été sauvegardé dans MLFlow. Cette démarche permet une gestion centralisée et simplifie son déploiement pour des utilisations futures.



SNN vs MobileNetV2



Shallow Neural Network

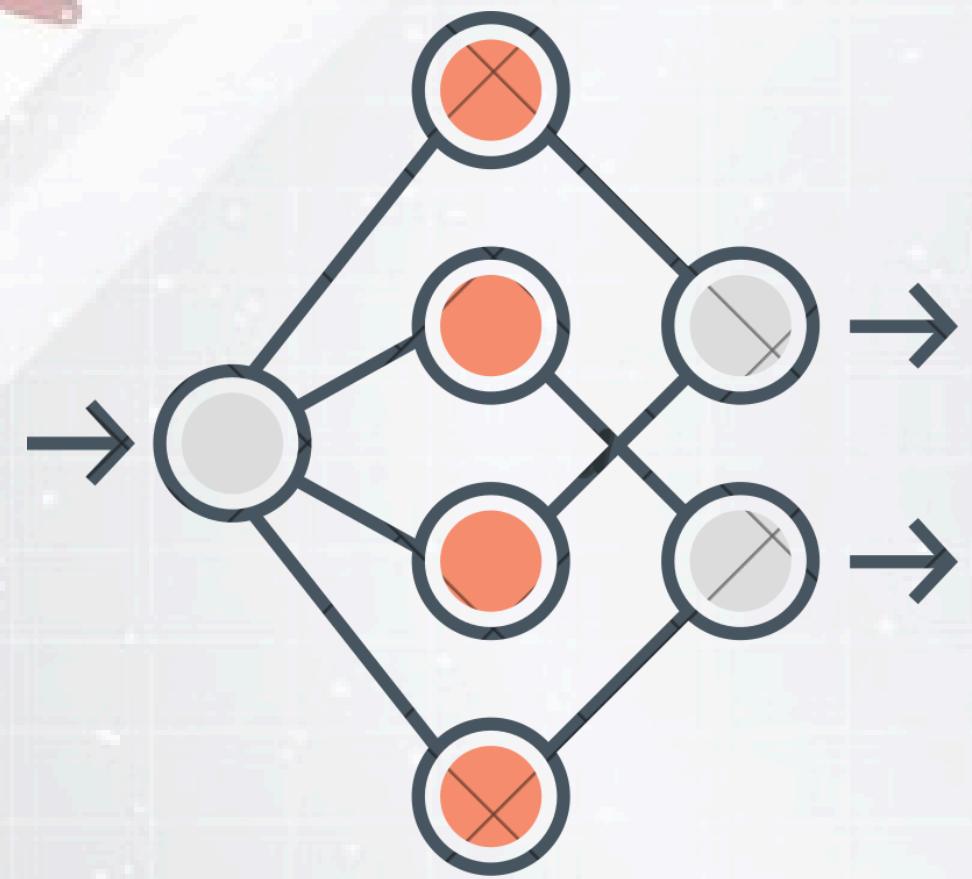


- Architecture simple et entièrement connectée.
- Conçue pour des expérimentations rapides et des ensembles de données de petite taille.
- Sert de base de comparaison avec des modèles plus complexes.

Entraînement

- Optimiseur : Adam.
- Fonction de Perte : Categorical Crossentropy.
- Métrique d'Évaluation : Accuracy.

MobileNetV2



- Modèle basé sur l'apprentissage par transfert (Transfer Learning).
- Utilisation de MobileNetV2 pré-entraîné sur ImageNet comme extracteur de caractéristiques.
- Ajout de couches personnalisées pour l'adaptation à la classification multiclasse

Entraînement

- Optimiseur : Adam.
- Fonction de Perte : Categorical Crossentropy.
- Métrique d'Évaluation : Accuracy.

MobileNetV2

Architecture

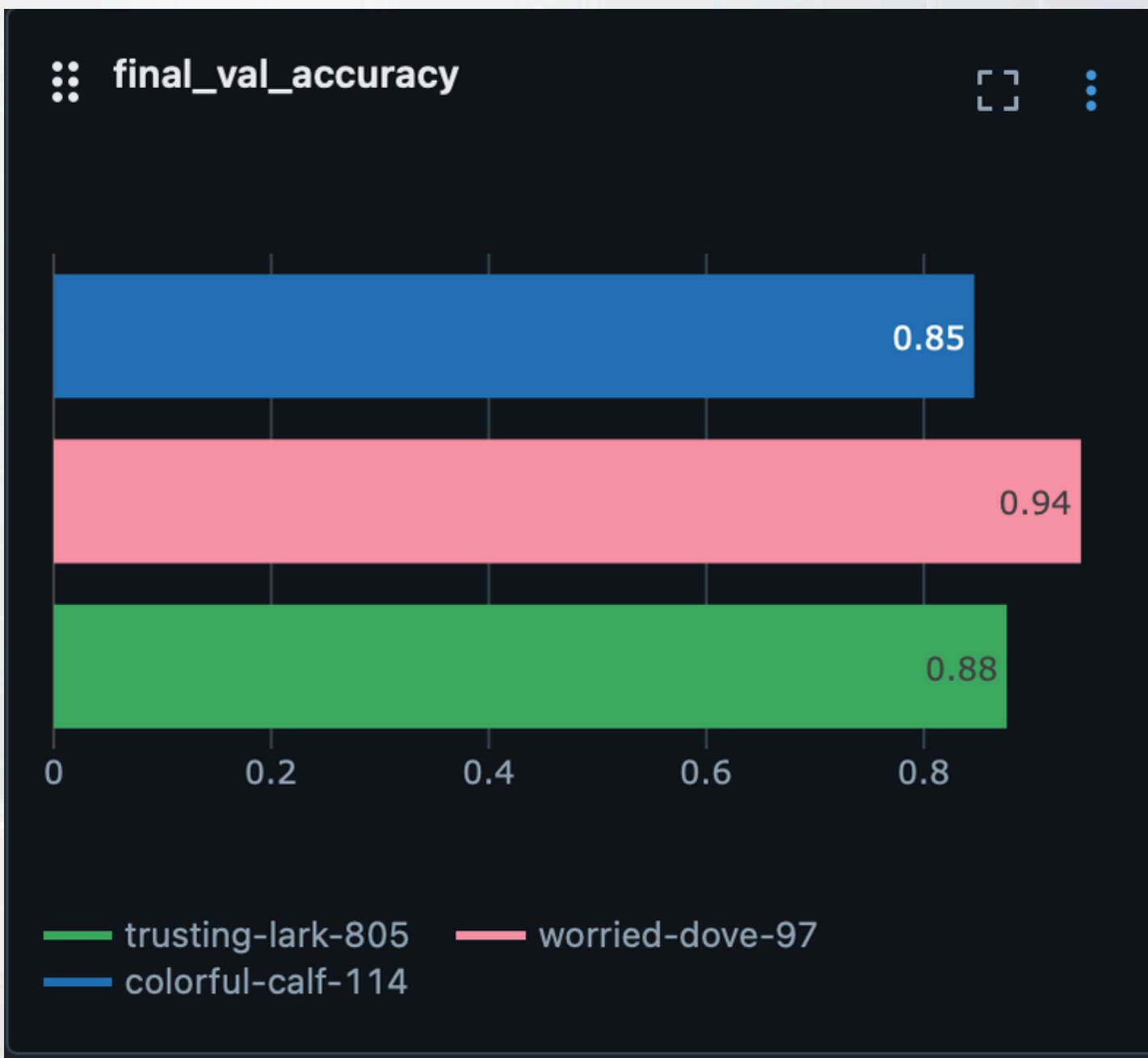
MobileNetV2 :

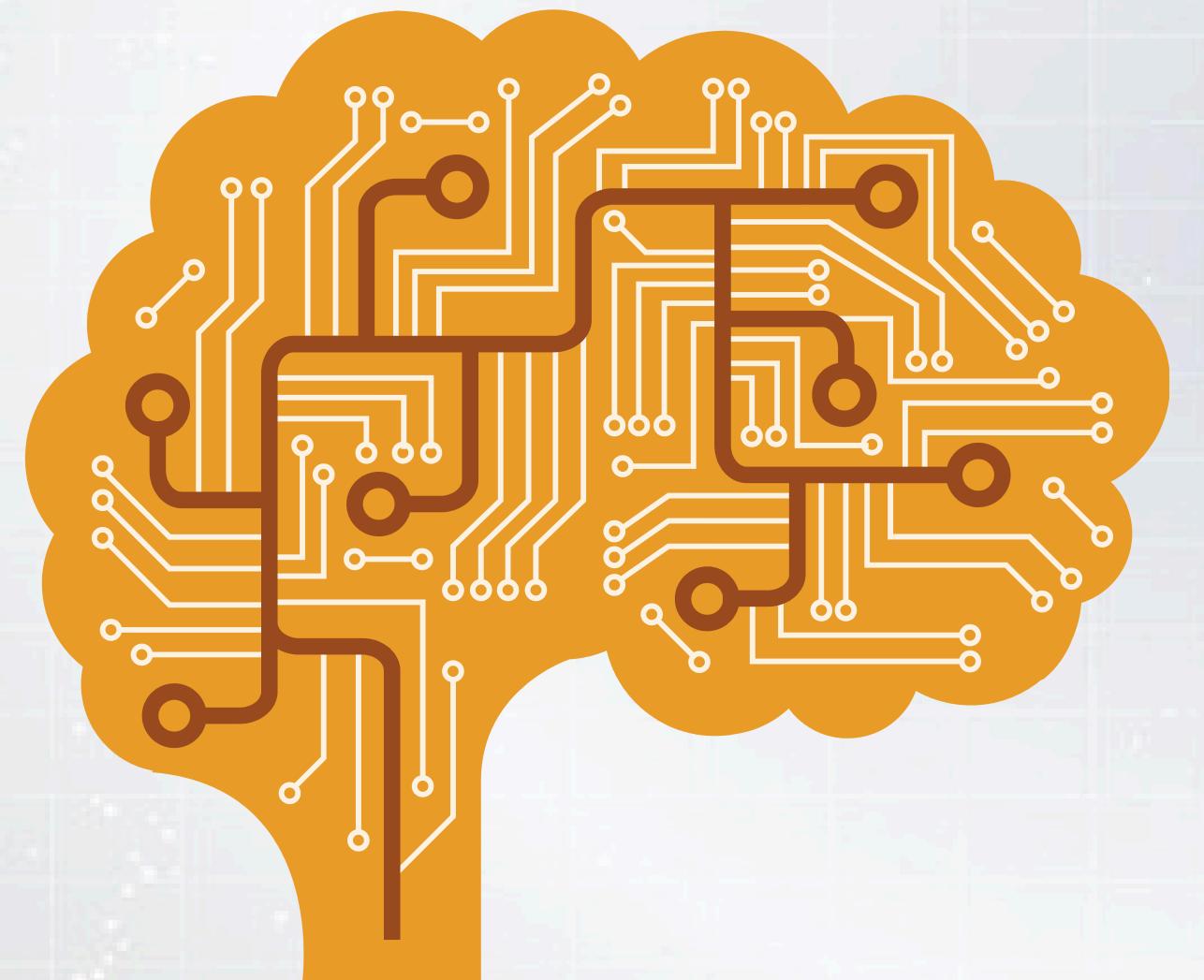
- Couche de base pré-entraînée, gelée (non modifiable) pour conserver les poids d'ImageNet.
- Fournit des caractéristiques riches et générales.

Couches Supplémentaires Personnalisées :

- GlobalAveragePooling2D pour aplatiser les caractéristiques.
- Deux couches denses avec activation ReLU.
- Dropout pour la régularisation (taux de 0,3).
- Couche dense finale avec activation softmax pour la classification.

MLflow log

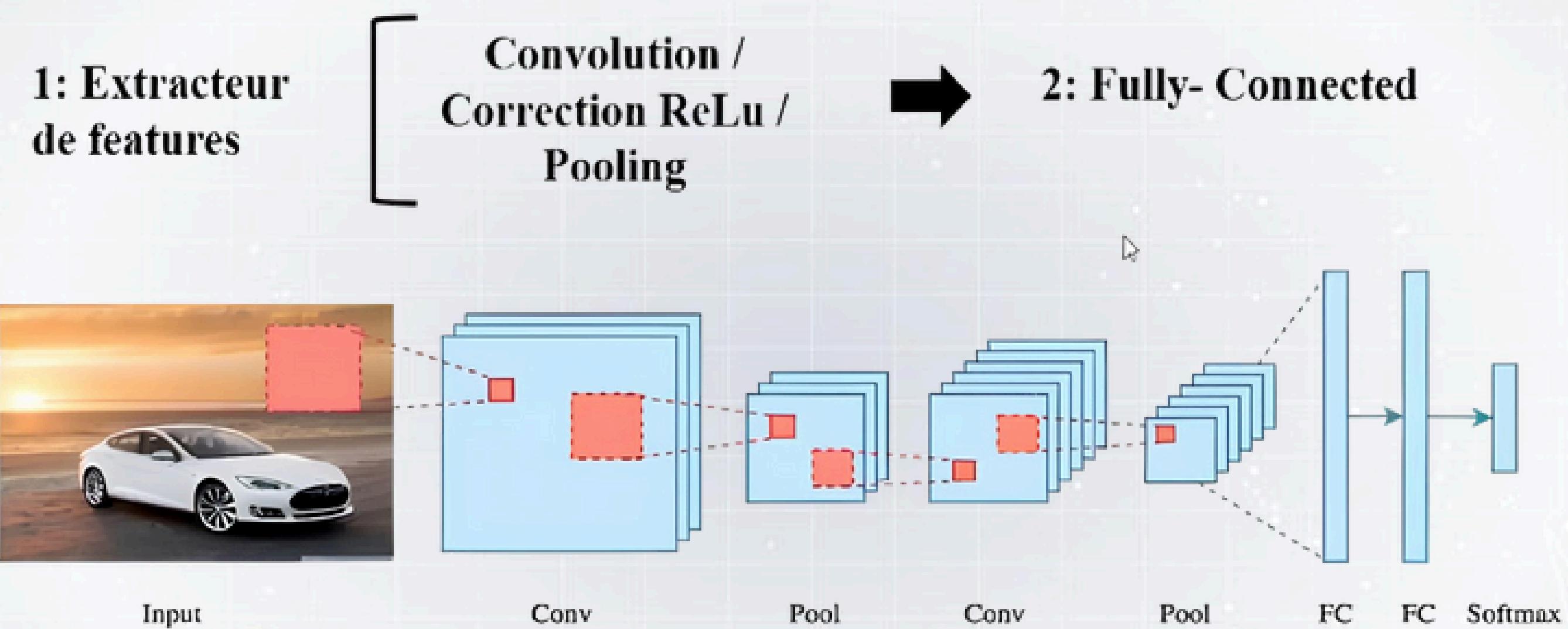




Convolutional Neural Network

Réseau de Neurones Convolutif

Les CNN sont spécialement conçus pour traiter des images en entrée



Réseau de Neurones Convolutif

L'architecture de CNN implémentée est comme suit :

- 1. Couches Convolutionnelles** : Le modèle commence par trois couches convolutionnelles avec des filtres croissants et des couches de pooling pour extraire et réduire les dimensions des caractéristiques de l'image.
- 2. Couches Entièrement Connectées** : Après l'aplatissement des caractéristiques, deux couches pleinement connectées sont ajoutées, avec une couche Dropout pour réduire le risque de surapprentissage.
- 3. Sortie et Compilation** : La sortie est une couche softmax pour la classification multi-classe, et le modèle est compilé avec des optimiseurs comme Adam, en utilisant des métriques telles que accuracy, recall et AUC.

Réseau de Neurones Convolutif

Paramètres à tester avec la méthode Grid_Search :

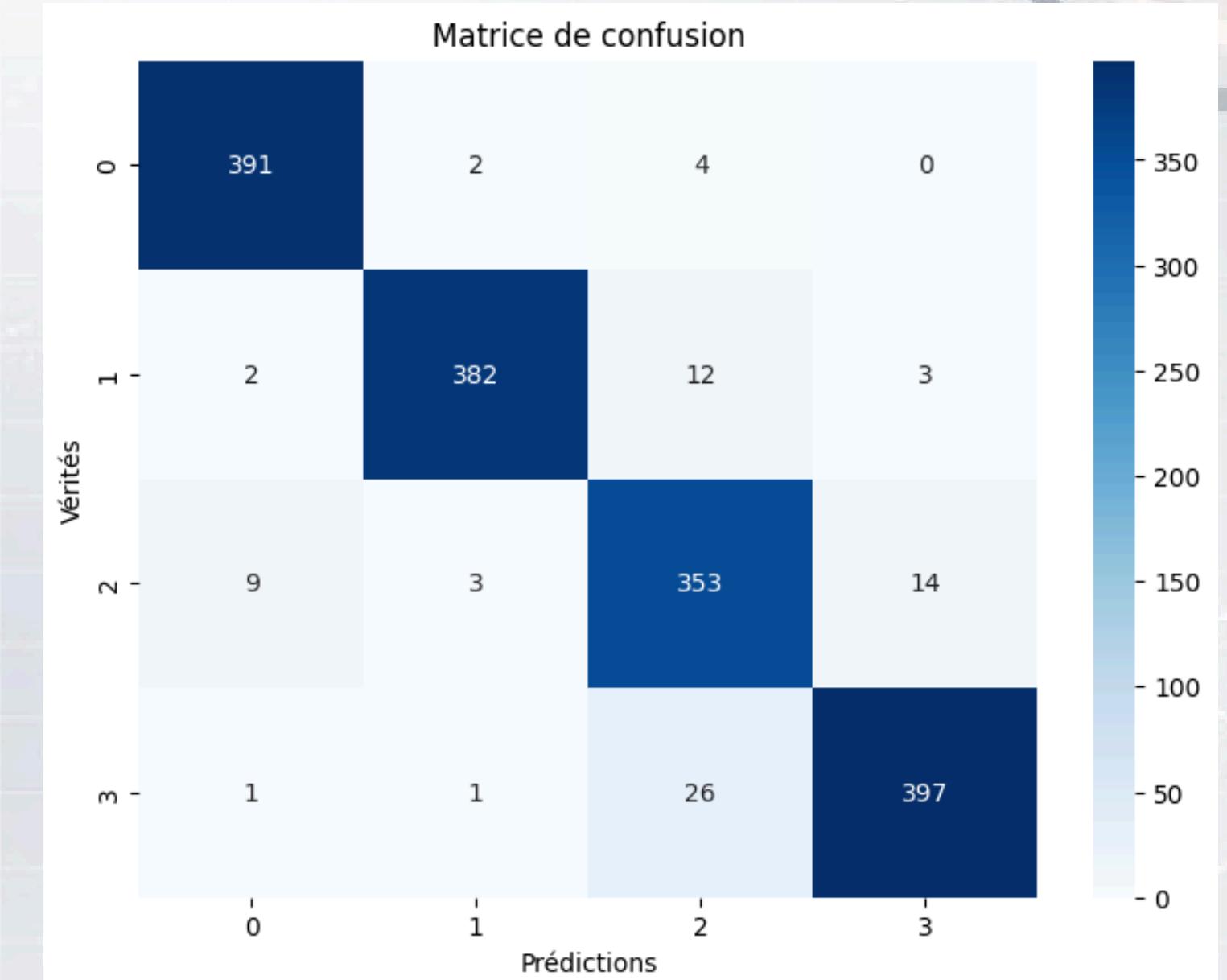
- **Optimiseur** : Adam, SGD, RMSprop.
- **Taux de Dropout** : 0.3, 0.5, 0.7.
- **Fonction d'activation** : ReLU, Tanh, Sigmoid.
- **Filtres** : 16, 32.
- **Taille du noyau de convolution** : (3, 3).
- **Taille des mini-batchs** : 16, 32.
- **Nombre d'époques** : 5, 10.

Durée du processus de Grid Search pour 108 combinaisons : 3 heures...

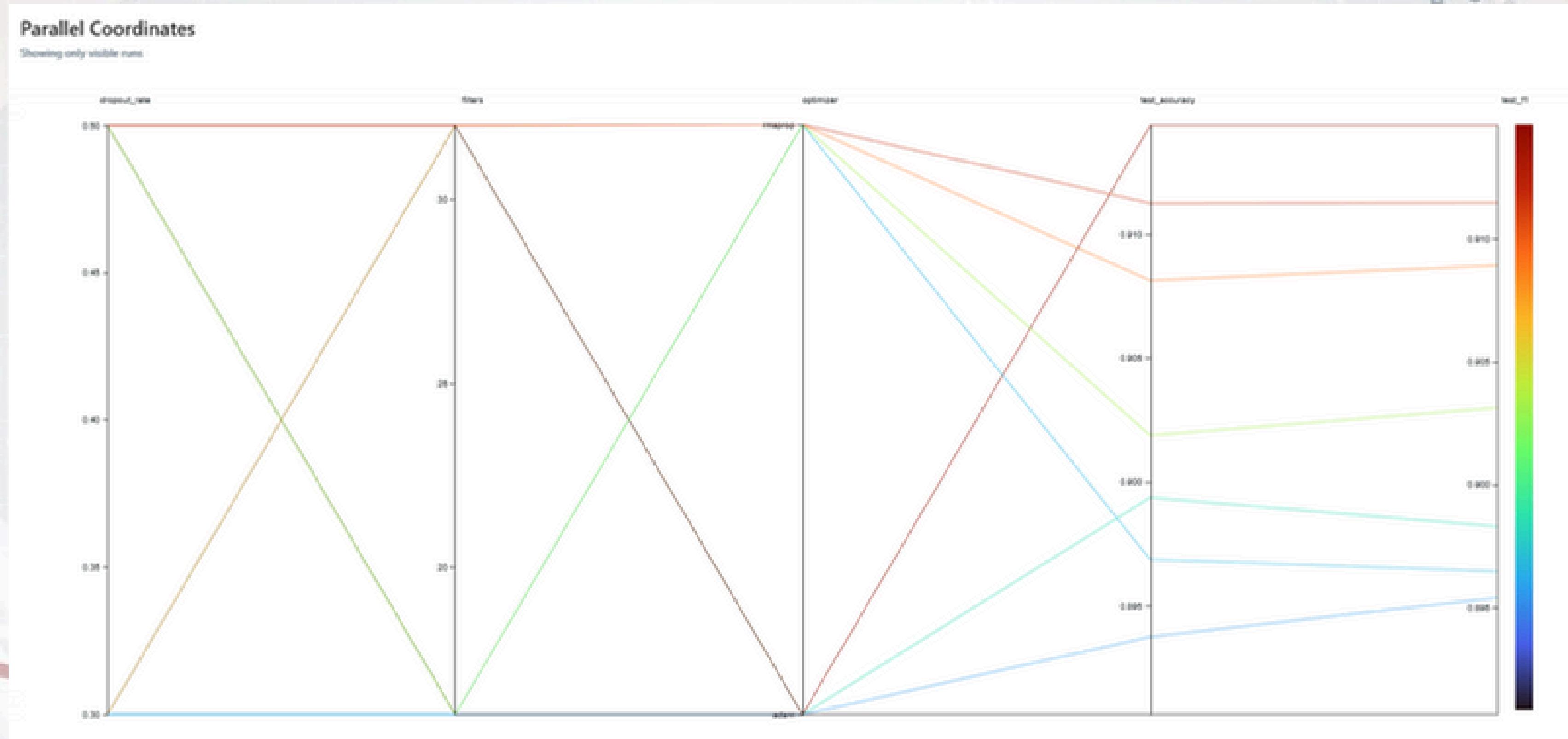
Réseau de Neurones Convolutif

La meilleure combinaison de paramètres (**Adam, Dropout 0.5, ReLU, 32 filtres, noyau (3, 3), batch size 16 et 10 époques**) a donné un score de **95.19%** sur le jeu de test, avec une perte de 0.1935 et un Recall de 95% .

Overfitting ?



Réseau de Neurones Convolutif





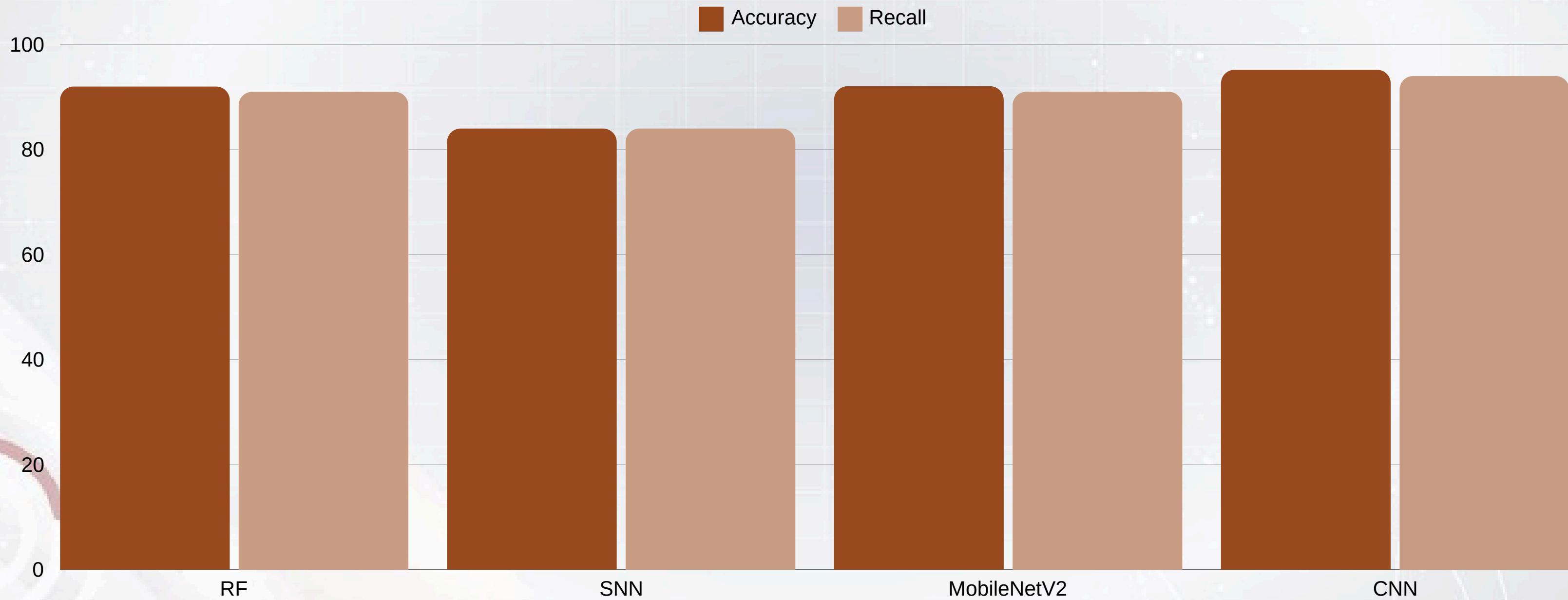
05

Résultats et comparaison

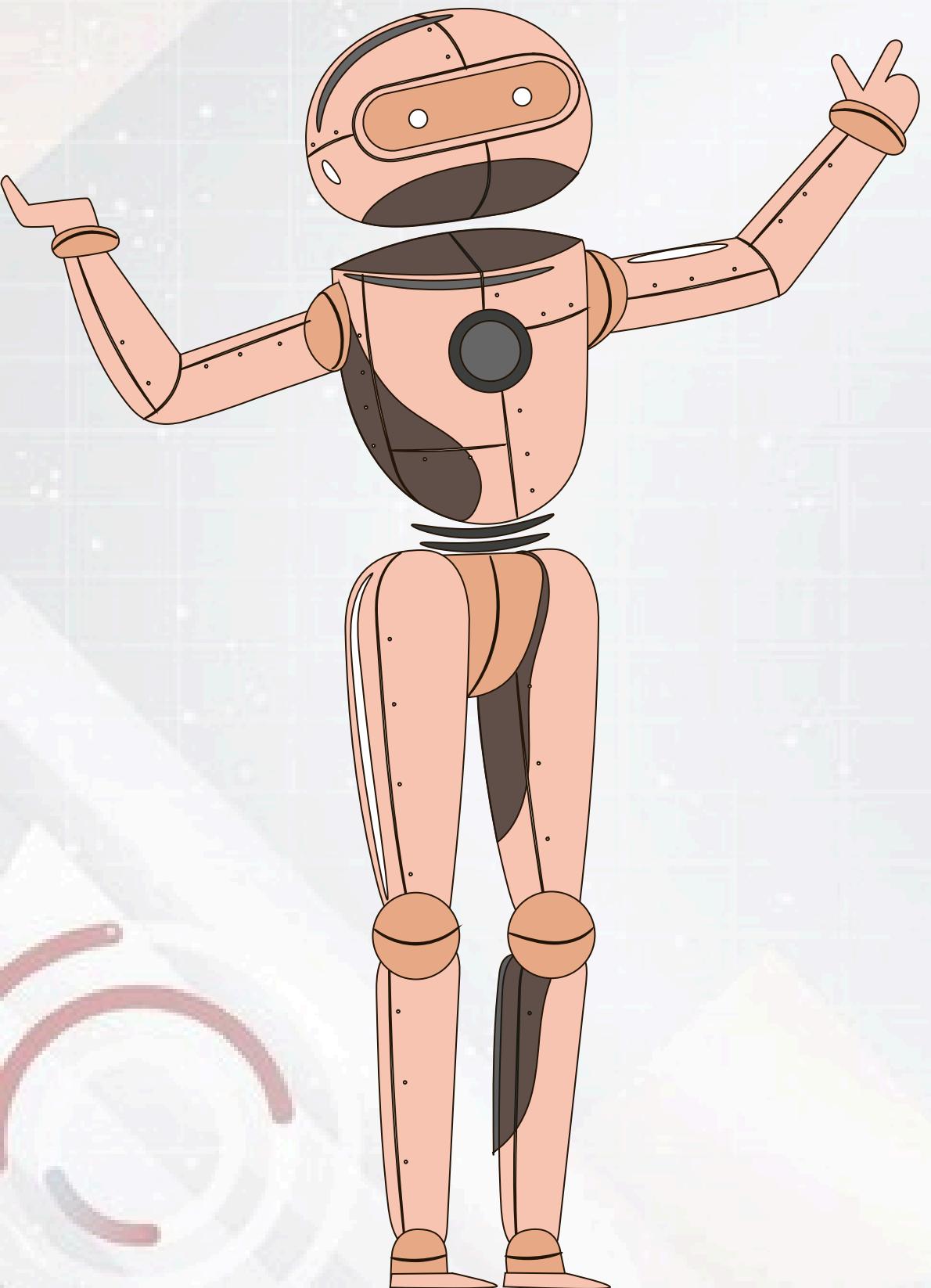
Résultats et Comparaison

Les trois modèles ont été évalués avec :

- Un **rapport de classification** pour la précision, le rappel et le F1-score.
- Une **matrice de confusion** pour analyser les erreurs de prédiction.
- Les courbes AUC-ROC pour mesurer les performances sur toutes les classes.



Résultats et Comparaison



- Le RF a étonnement de **bonnes performances** par rapport au temps d'apprentissage pour des données de type image.
- Le réseau de neurones simple était le **plus rapide** mais **moins performant** sur les motifs complexes.
- Le MobileNetV2 a offert le **meilleur équilibre** entre rapidité et précision.
- Le CNN personnalisé a **excellé** sur des données suffisantes, mais au prix d'un temps d'entraînement **plus long et lent**.

06

Explication Model



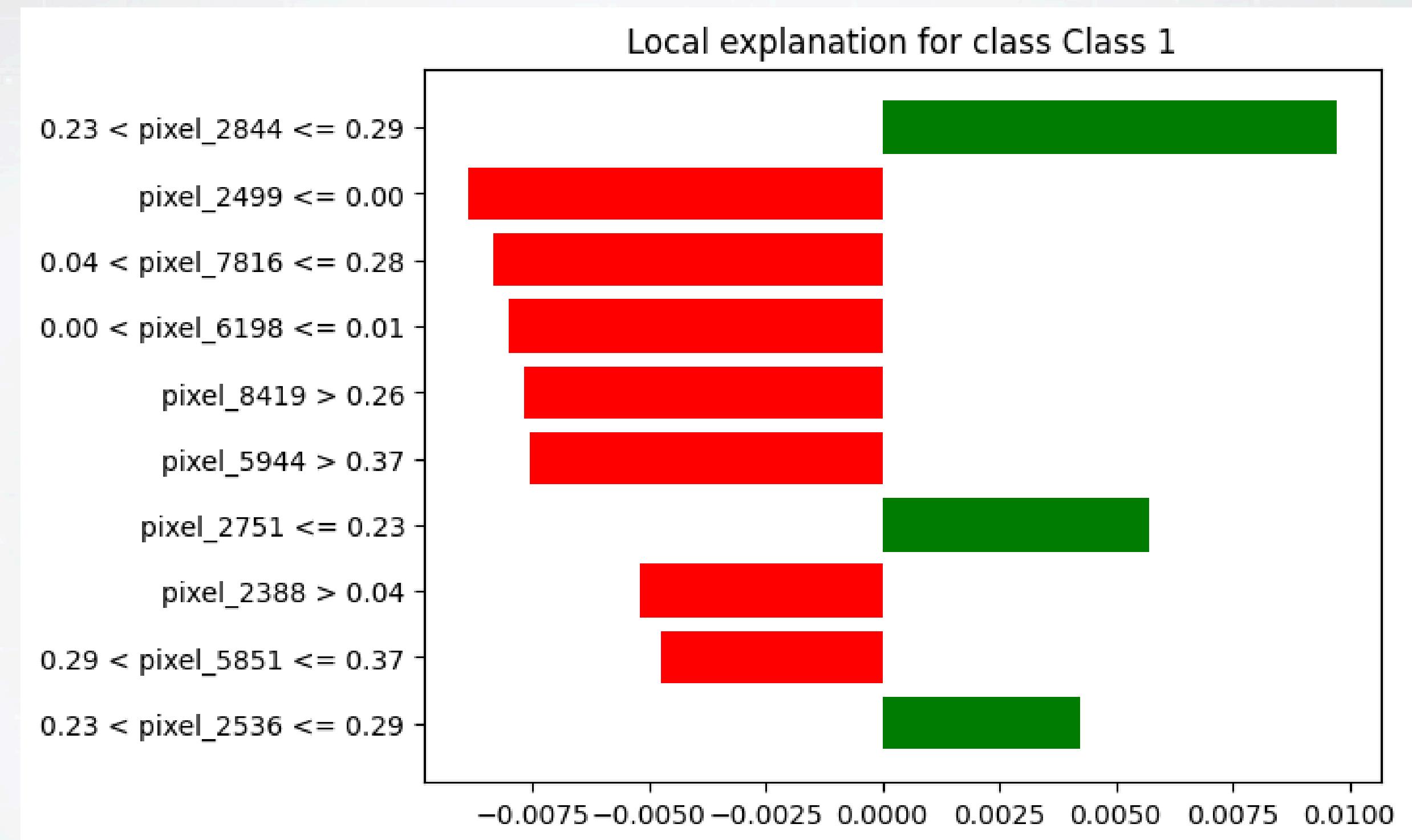
LIME

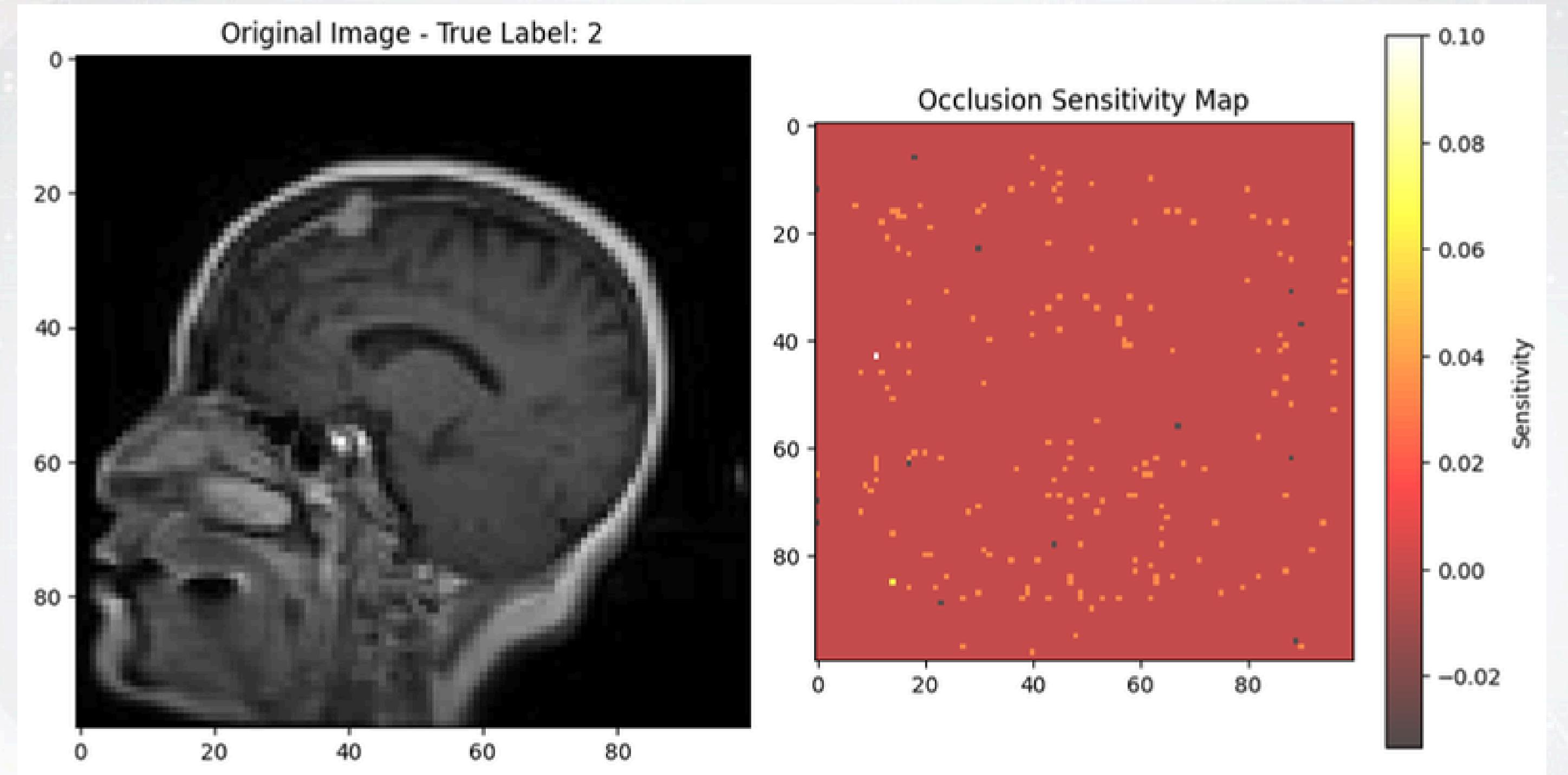
Prediction probabilities	NOT Class 2	Class 2	NOT Class 0	Class 0	NOT Class 1	Class 1	NOT Class 3	Class 3
Class 0 0.23			0.00 < pixel_6964 < 0.01		pixel_3131 < 0.23	pixel_2499 < 0.00	pixel_5156 > 0.16	pixel_1533 > 0.35
Class 1 0.07			pixel_4270 > 0.36		pixel_252 > 0.07	0.04 < pixel_7836 < 0.05	pixel_8461 > 0.36	
Class 2 0.47			pixel_3891 > 0.06		pixel_7789 > 0.06	0.00 < pixel_6198 < 0.01	pixel_5502 < 0.23	
Class 3 0.03			pixel_117 < 0.09		pixel_8189 > 0.38	pixel_3419 > 0.26	0.01 < pixel_7448 < 0.05	
			pixel_8688 > 0.34		pixel_241 > 0.05	pixel_5944 > 0.37	pixel_6050 > 0.17	
			pixel_2967 > 0.36			pixel_2751 < 0.23	0.04 < pixel_1338 < 0.08	
			pixel_3754 < 0.22			pixel_2388 > 0.04	pixel_2388 > 0.04	
			pixel_816 < 0.01			0.29 < pixel_3851 < 0.30	pixel_767 > 0.14	
			pixel_1483 > 0.03			0.23 < pixel_2334 < 0.26		
			pixel_9801 < 0.00		0.24 < pixel_3080 < 0.25			

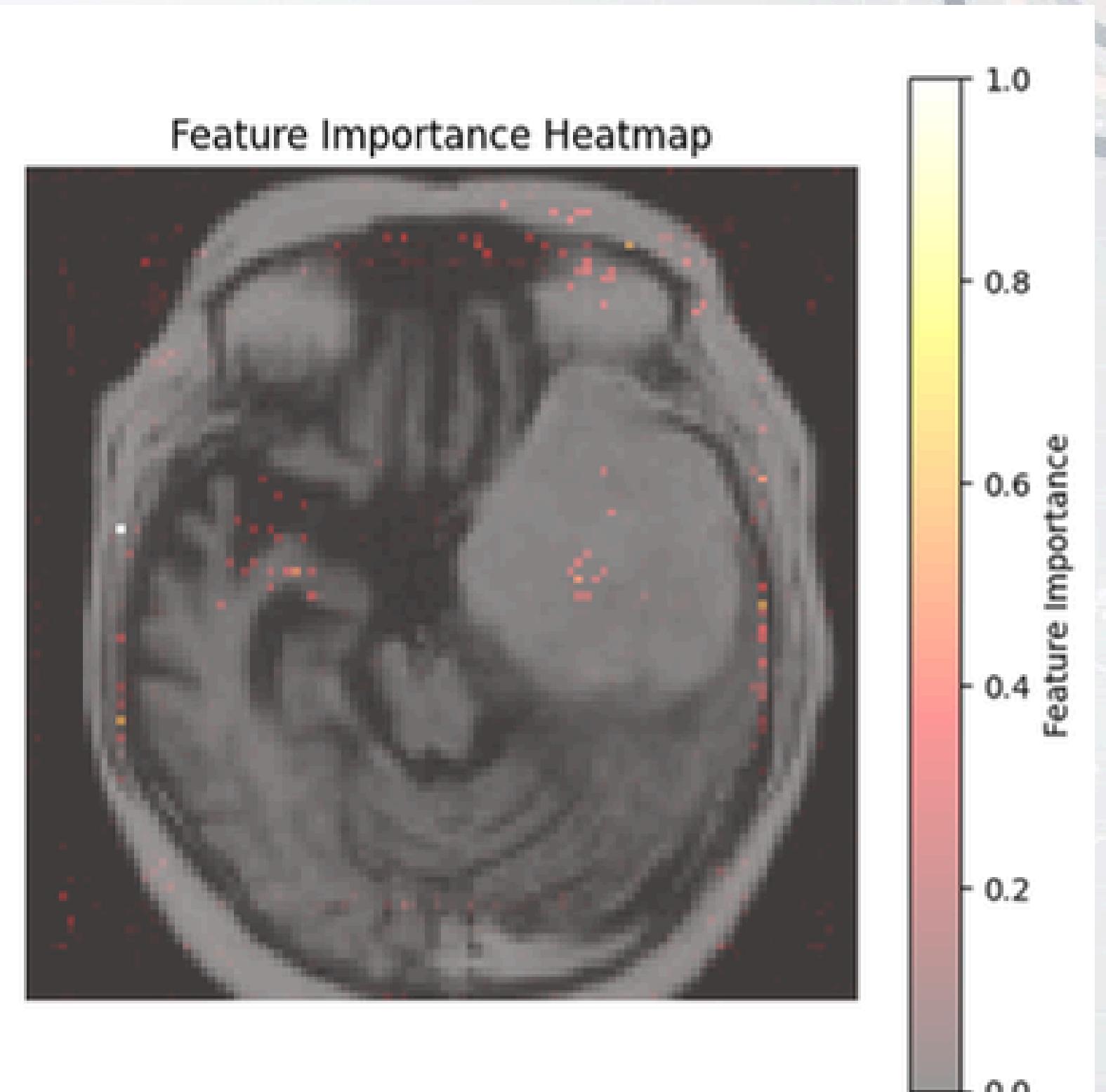
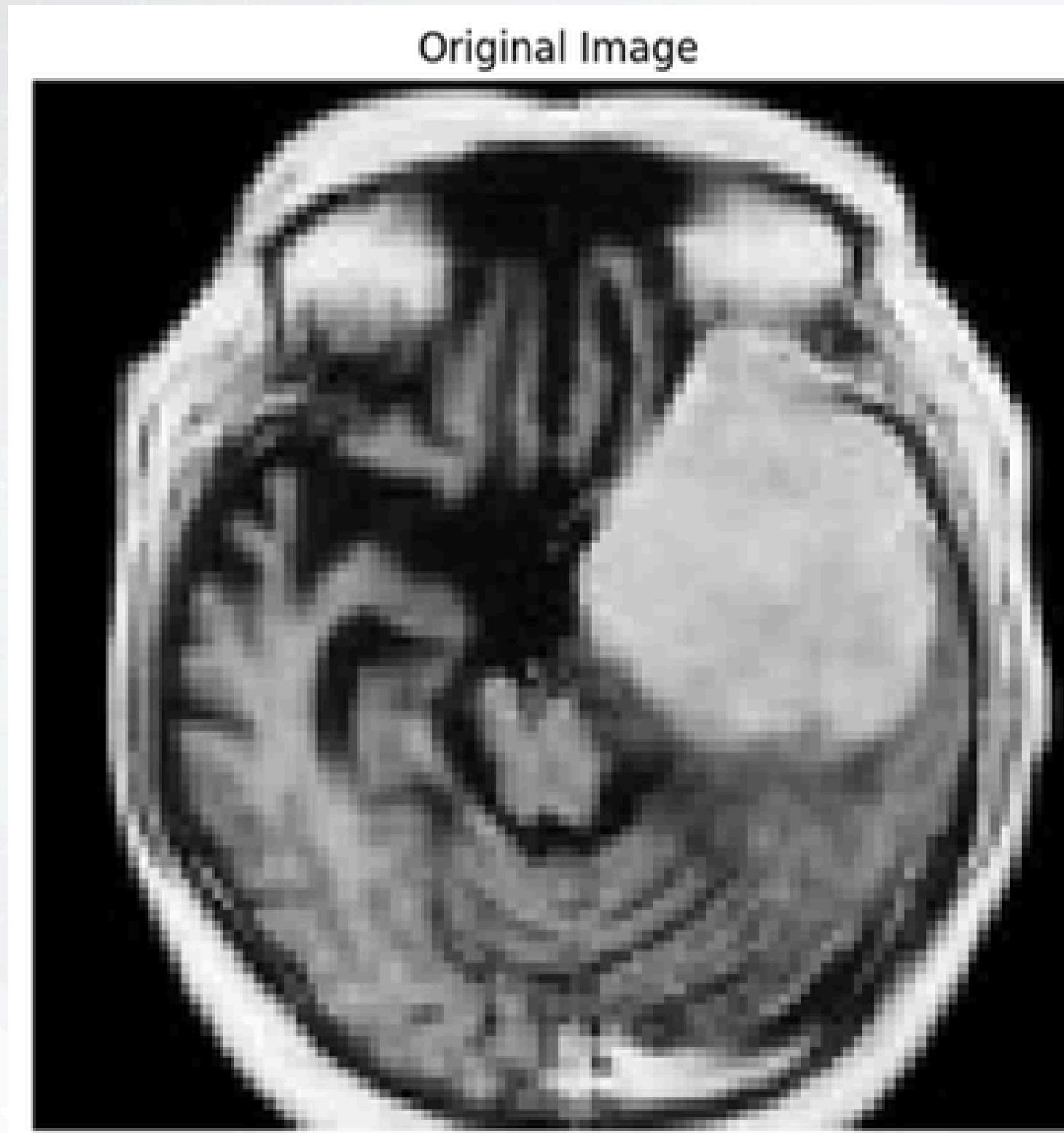
Feature Value

pixel_17420.12
pixel_42700.80
pixel_38910.82
pixel_117-0.09
pixel_86880.35
pixel_29670.76
pixel_37540.12
pixel_816-0.00
pixel_14830.59
pixel_98010.00

LIME







07

Application Web



Application Web

Après avoir sélectionné le meilleur modèle, nous allons l'intégrer dans une application web. Cette application permettra aux utilisateurs de soumettre leurs données et d'obtenir des prédictions en temps réel, rendant notre solution accessible et facile à utiliser.

```
(Tp1) PS C:\Cours\Processus Data\TP_Environment_Setup> export MLFLOW_TRACKING_URI=http://localhost:5000
export : The term 'export' is not recognized as the name of a cmdlet, function, script file, or operable program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ export MLFLOW_TRACKING_URI=http://localhost:5000
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (export:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

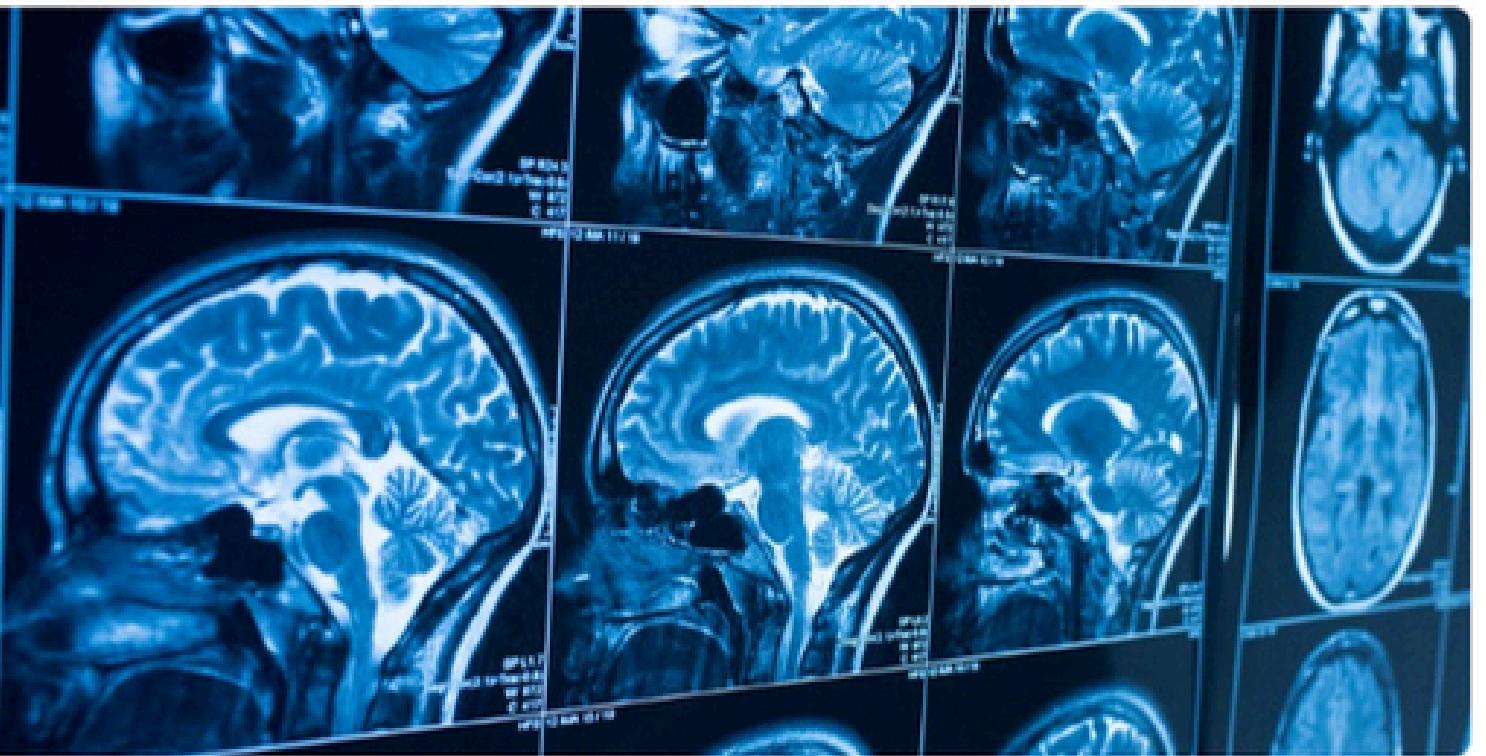
(Tp1) PS C:\Cours\Processus Data\TP_Environment_Setup> $env:MLFLOW_TRACKING_URI="http://localhost:5000"
(Tp1) PS C:\Cours\Processus Data\TP_Environment_Setup>
(Tp1) PS C:\Cours\Processus Data\TP_Environment_Setup>
(Tp1) PS C:\Cours\Processus Data\TP_Environment_Setup>
(Tp1) PS C:\Cours\Processus Data\TP_Environment_Setup> mlflow models serve -m "models:/BrainTumorClassifierBestModel/1"
--no-conda -p 5001
Downloading artifacts: 100%|██████████| 7/7 [00:02<00:00, 3.14it/s]
2024/12/11 04:10:01 INFO mlflow.models.flavor_backend_registry: Selected backend for flavor 'python_function'
Downloading artifacts: 100%|██████████| 7/7 [00:02<00:00, 2.75it/s]
2024/12/11 04:10:06 INFO mlflow.pyfunc.backend: === Running command 'waitress-serve --host=127.0.0.1 --port=5001 --ident
=mlflow.mlflow.pyfunc.scoring_server.wsgi:app'
INFO:waitress:Serving on http://127.0.0.1:5001
2024/12/11 11:24:05 WARNING mlflow.pyfunc.scoring_server: If using 'instances' as input key, we internally convert the d
ata type from 'records' (List[Dict]) type to 'list' (Dict[str, List]) type if the data is a pandas dataframe representat
ion. This might cause schema changes. Please use 'inputs' to avoid this conversion.

2024/12/11 11:27:04 WARNING mlflow.pyfunc.scoring_server: If using 'instances' as input key, we internally convert the d
ata type from 'records' (List[Dict]) type to 'list' (Dict[str, List]) type if the data is a pandas dataframe representat
ion. This might cause schema changes. Please use 'inputs' to avoid this conversion.
```

Application Web

Nous avons utilisé le framework Flask pour développer une application web simple et intuitive, offrant une expérience utilisateur fluide et accessible.

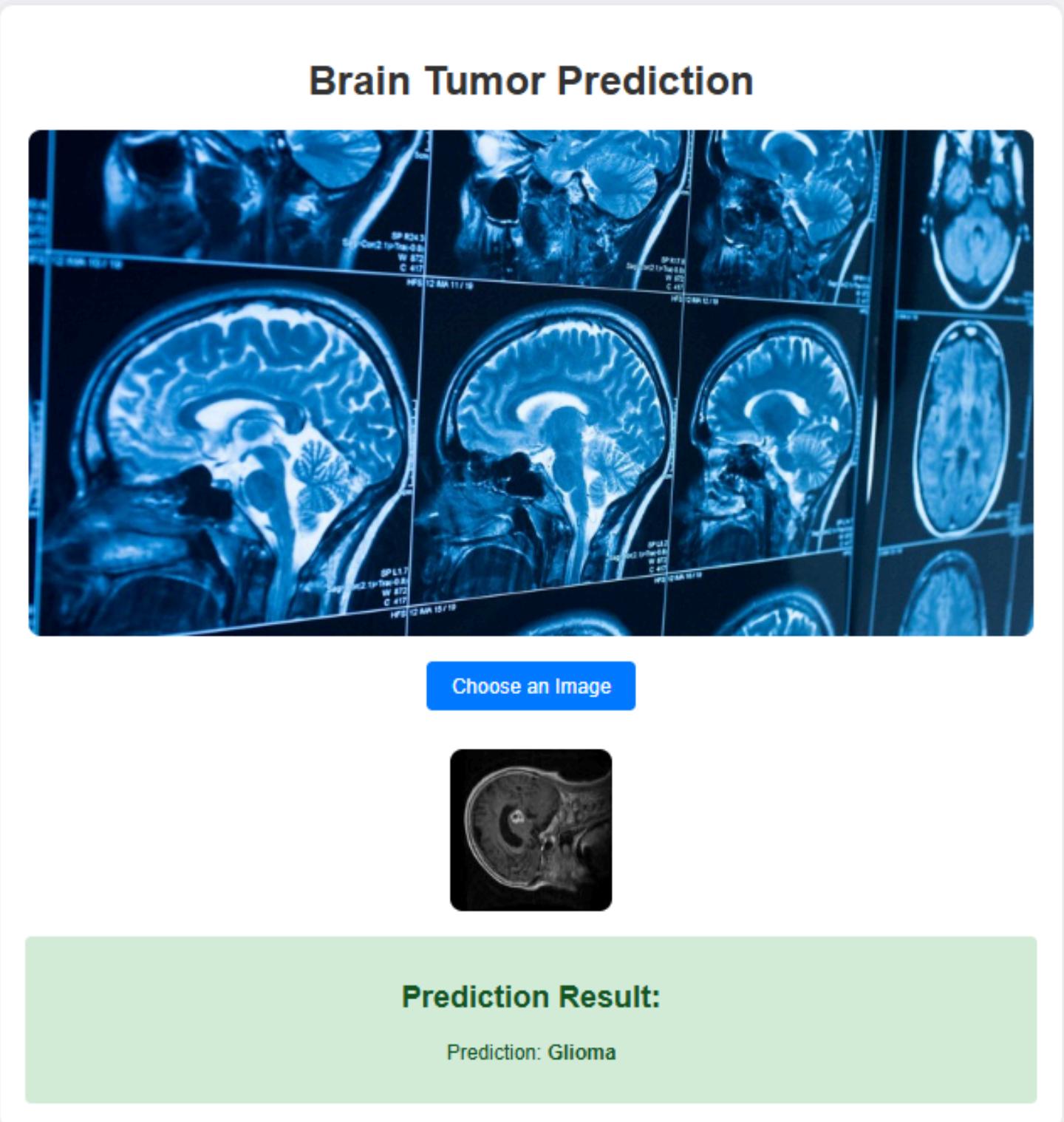
Brain Tumor Prediction



Choose an Image

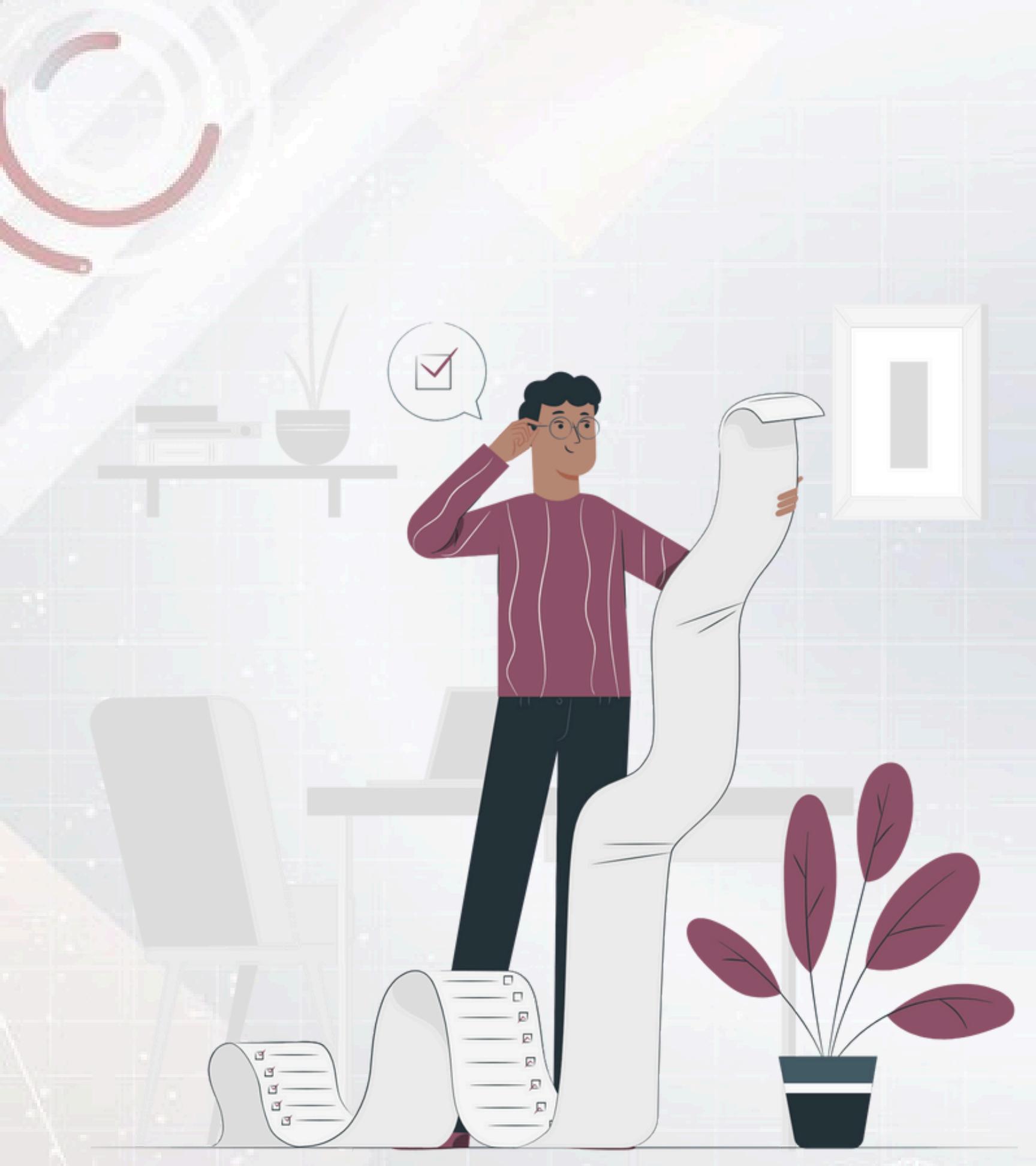
Application Web

Après que l'utilisateur télécharge son image, celle-ci est traitée pour extraire les paramètres nécessaires. Ces derniers sont ensuite transmis à l'API pour effectuer les calculs, et le résultat est affiché directement sur notre page web.



08

CONCLUSION

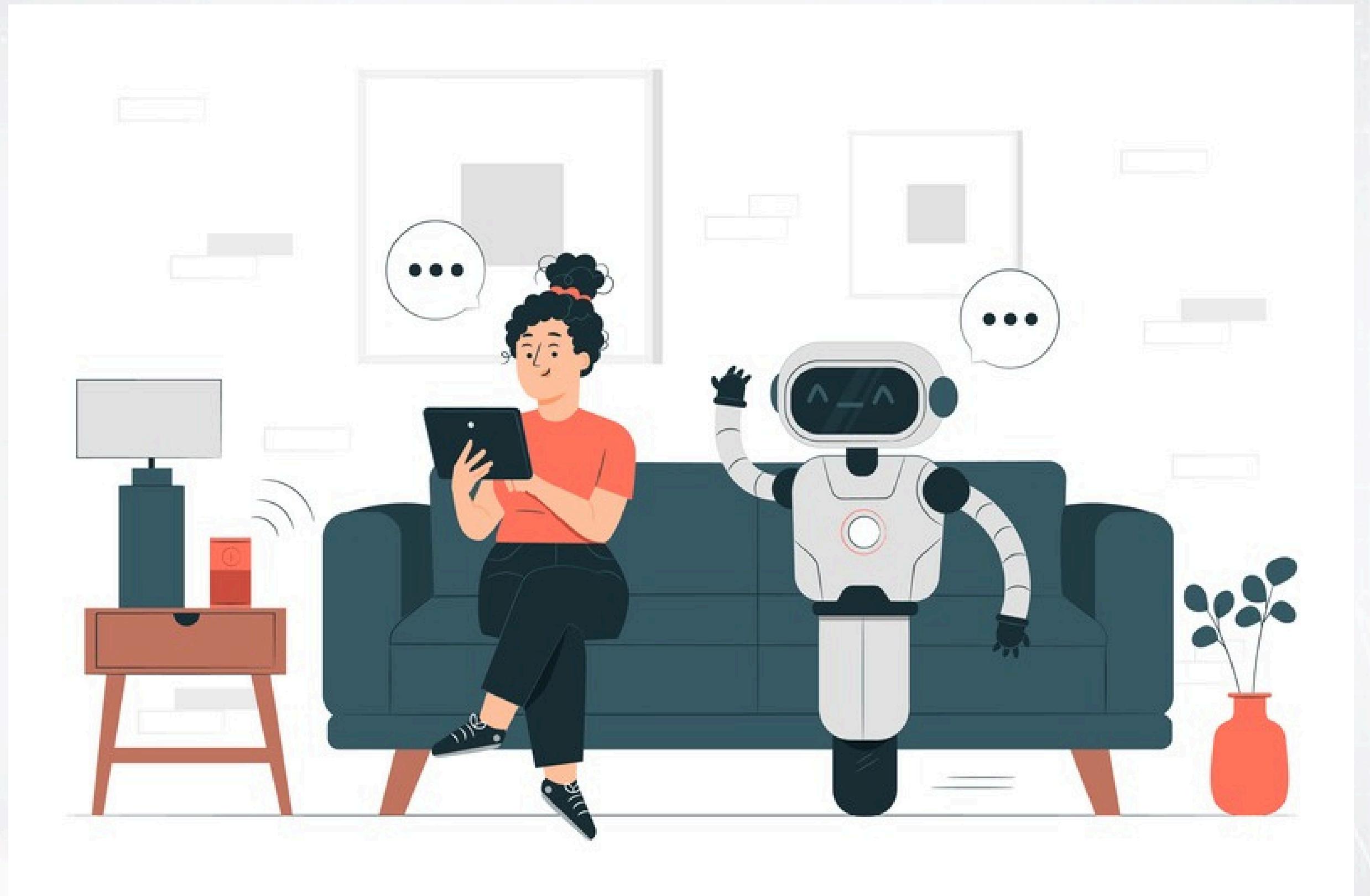


Conclusion

Bien que les modèles de deep learning comme MobileNetV2 ou CNN offrent un bon compromis entre rapidité et précision, le Random Forest a étonnement montré de bonnes performances, soulignant l'importance du choix du modèle selon le compromis entre précision et temps de calcul.



Merci pour votre attention.





QUESTIONS