

Projet Reconnaissance de plantes

et de leurs maladies

\_\_\_\_\_\_\_\_\_

Cahier des charges projet MLOps

## Contexte et Objectifs

L’application doit permettre aux jardiniers amateurs et/ou professionnels d’identifier facilement les plantes de leur jardin ainsi que de diagnostiquer leurs éventuelles maladies. La version initiale comportera l’identification de 14 types de plantes ainsi que de 26 types de maladies différentes, soit 38 types de catégories différentes (càd. un couple « type de maladie ou feuille saine » pour un « type de plante » spécifque).

Le commanditaire de l’application est le porte-parole d’un regroupement d’agriculteurs d’Afrique de l’Est (Climat tropical) subissant le réchauffement climatique et donc une augmentation des maladies et de leurs types dans leurs cultures.

Les utilisateurs de l’application seront les agriculteurs eux-mêmes et certaines coopératives (via un « compte coopérative » dédié à chaque coopérative).

Les administrateurs de l’application seront les concepteurs de la solution (sans développement de l’application elle-même).

L’application devra s’intégrer dans un site web dédié ou dans une application pour smartphone ou tablette. Elle sera utilisée via une interface graphique qui communiquera avec l’API dédiée.

La fonctionnement de base de la solution sera le suivant : un utilisateur pourra envoyer une photo de feuille de plante à l’API et la solution prédira le type de plante, si la feuille est saine ou malade, ainsi que la maladie associée le cas échéant.

Dans la première version proposée, il n’y aura pas de possibilité d’envoi multiple de photos. Cependant, en fonction du temps qu’il nous restera lors du développement, cela pourrait être ajouté.

## Modèle

Le modèle retenu est le modèle **EfficientNetV2S avec 6 couches de sortie personnalisées**.

Le modèle EfficientNetV2S est basé sur une architecture de réseaux de neurones convolutifs (CNN) appelée EfficientNet. L'architecture EfficientNet a été conçue pour être à la fois plus précise et plus efficace en termes de calcul par rapport à de nombreuses autres architectures de réseaux neuronaux.

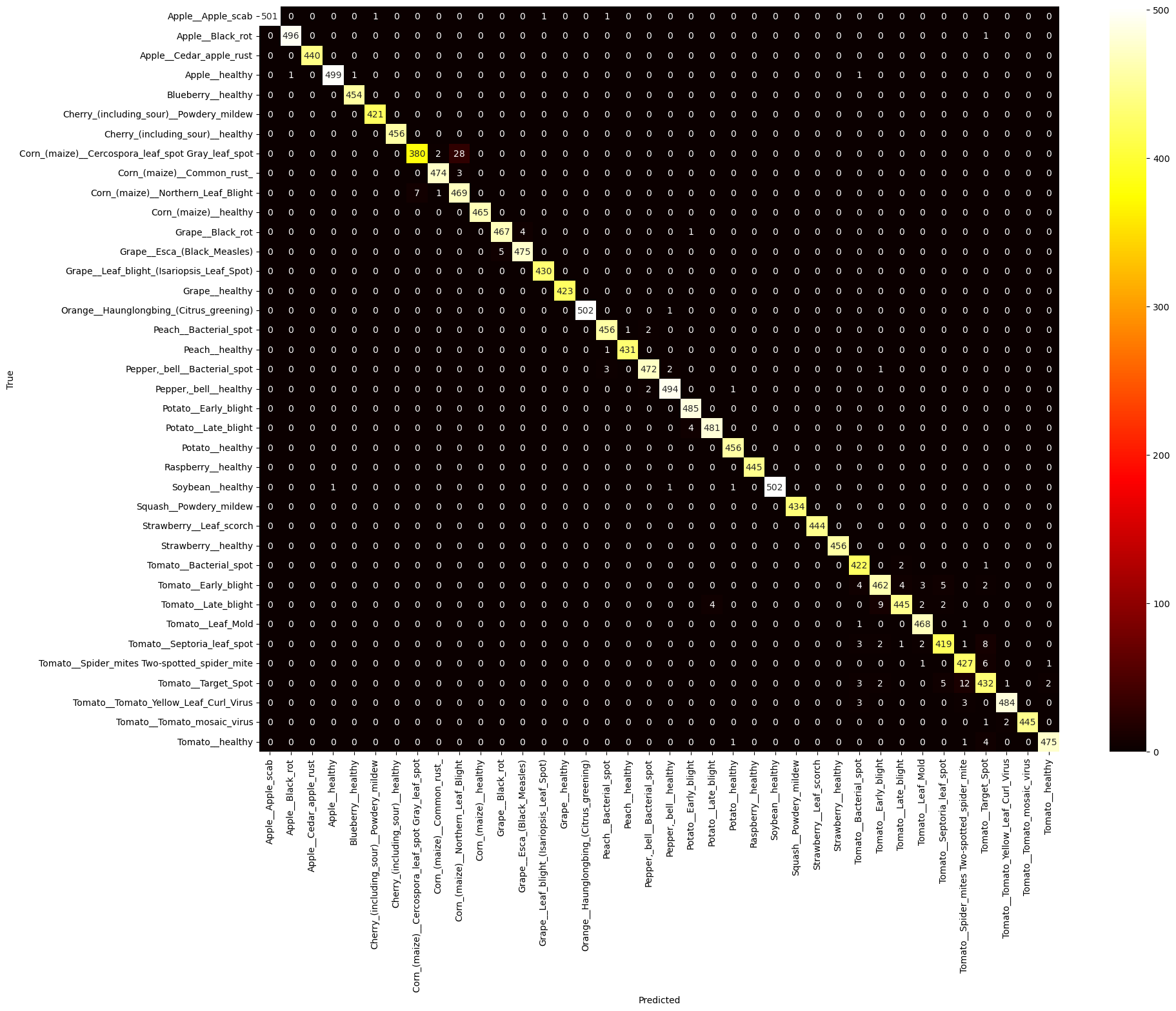
Ce modèle est une version optimisée de l'architecture EfficientNet spécifique à des systèmes ayant des ressources limitées, ce qui signifie qu'elle est plus légère et plus rapide que d'autres variantes. Elle est conçue pour être utilisée dans des environnements où la puissance de calcul ou la mémoire sont limitées.

Le modèle a 38 classes à prédire dans la première version. Si le temps le permet, une classe « Inconnu » sera ajoutée. Dans ce cas, le modèle prédirait donc 39 classes.

La préparation des données des données d’entraînement a été faite en utilisant le « preprocessing » fourni avec le modèle EfficientNetV2S avec ajout de bruit supplémentaire afin d’éviter le sur-apprentissage autant que possible. Concernant les données de validation, en plus du même « preprocessing », certaines modifications ont été apportées afin que les images soient le plus proche possible de ce que l’on pourrait trouver dans la nature.

L’entraînement du modèle a été effectué en variant les tests, avec ou sans modifications d’image (ajout de bruit, etc.), et également par une étude de la variation des hyper paramètres via un « Grid Search ».

Vous trouverez, ci-dessous, une HeatMap de la matrice de confusion liée au modèle choisi :



Les métriques d’évaluation de la prédiction seront les suivants :

* Les métriques techniques réelles de la prédiction :
  + Accuracy de la prédiction
  + Temps de prédiction
* La satisfaction du client sur :
  + La prédiction effectuée
  + Le temps de réponse de la prédiction

## Bases de données

Notre API sera en interaction avec 5 bases de données différentes.

### Base de données des images des plantes : NPD.csv

Cette base de données est au format CSV et reflète l’entièreté du Dataset « New Plant Diseases » qui contient plus de 87000 images de plantes, saines ou malades, divisées de façon homogène par catégorie (càd. par couple plante/maladie). Cette base de données ne contient pas directement les images mais les chemins d’accès aux images ainsi que les caractéristiques associées à chacune (type de plante, type de maladie, jeu de test/validation). De ce fait, l’arborescence des images « physiques » est également nécessaire pour pouvoir utiliser les images.

Les images de cette base de données sont réparties en deux jeux : entraînement et validation. Le jeu d’entraînement comporte 70 292 images et celui de validation 17571 images.

Les images sont réparties en 38 classes, correspondant à un type de maladie pour un type de plante. Si la plante est saine, le type de maladie sera nommé « healthy » (saine). Comme indiqué, la répartition des images est uniforme pour ces 38 classes.

La structure de la base NPD.csv est la suivante :

* Categorie : type de plante et type de maladie
* Plante : type de plante
* Maladie : type de maladie
* Saine : vaut 0 si la feuille est malade, 1 si saine
* Set : jeu de données associé, càd. « train » ou « valid »
* DirPath : répertoire où est stocké l’image
* FileName : nom de l’image
* FilePath : chemin complet de l’image

### Base de données de « démonstration » des images des plantes : NPD\_Mini.csv

Cette base de données est au format CSV et est une version « miniaturisée » du Dataset « New Plant Diseases ». Elle a été créée dans un but de démonstration et ne contient qu’une seule image de plante par catégorie (càd. par couple plante/maladie). Nous ne retrouverons donc dans cette base de donnée et le Dataset associé que 38 images d’entraînement et 38 images de validation.

La structure de la base « NPD\_Mini.csv » est identique à celle de la base principale « NDP.csv ».

### Base de données tampon : NPD\_Buffer.csv

Cette base de données sera utilisée pour stocker les caractéristiques des photos envoyées par les utilisateurs pour être potentiellement ajoutées à la base principale NPD.csv (et le Dataset associé). Les photos de cette base sont analysées manuellement afin de s’assurer de la bonne qualité de l’image et pour vérifier la cohérence du type de maladie et de plante attribués par l’utilisateur lors de la demande d’ajout.

La structure de la base est la même que celle de la base principale « NPD.csv » à laquelle a été ajouté un nouveau champ, « statut ». Celui-ci peut avoir les valeurs suivantes :

* 0 : image à vérifier
* 1 : image vérifiée à supprimer (incohérences détectées)
* 2 : image vérifiée à intégrer dans la base NPD.csv

### Base de données utilisateurs : Users.csv

Cette base de données permet de lister, pour chaque utilisateur, son mot de passe, son adresse email, ainsi que son type d’accès.

Cette base de données est au format CSV et n’est pas sécurisée.

La structure de cette base sera la suivante :

* Username : nom de connexion de l’utilisateur
* Firstname : prénom de l’utilisateur
* Lastname : nom de famille de l’utilisateur
* Email : adresse email de l’utilisateur
* Password : mot de passe de l’utilisateur
* Access\_Type : type d’accès des utilisateurs, càd. « admin » ou « user »

### Base de données de logs et de Monitoring : api\_log.json

Cette base de données sera utilisée pour stocker toutes les informations de connexion ou d’utilisation de l’API. Elle a pour but de tracer toutes les activités et sera également nécessaire pour afficher l’historique des actions d’un utilisateur.

Cette base de données est au format JSON et est non sécurisée.

La structure de cette base est de la forme suivante :

* Logtime : date et heure d’utilisation
* Username : nom de connexion de l’utilisateur
* Endpoint : nom de la route utilisée
* Endpoint\_Params : liste des paramètres utilisés sur la route
* Response\_Code : code statut de la route utilisée
* Image\_Name : nom de la photo utilisée pour la prédiction
* Image\_Path : chemin de la photo utilisée pour la prédiction
* Prediction : nom de la classe prédite
* Accuracy : taux de précision de la prédiction
* Pred\_Duration : temps de réponse du calcul de la prédiction
* Satis\_Pred : satisfaction de l’utilisateur du résultat de la prédiction
* Satis\_Pred\_Duration : satisfaction de l’utilisateur du temps de réponse pour la prédiction

## API

Liste des routes prévues pour l’API :

* Route de vérification du statut de l’API sans authentification
  + Route :
    - <http://127.0.0.1:8080/apichk> (GET)
  + Paramètre(s) :
    - Aucun
  + Sortie :
    - Message de confirmation du statut de l’API
  + Exemple :
    - curl.exe -X GET http://127.0.0.1:8080/apichk
* Route de vérification du statut de l’API avec authentification
  + Route :
    - <http://127.0.0.1:8080/secapichk> (GET)
  + Paramètre(s) :
    - Nom de connexion de l’utilisateur
    - Mot de passe
  + Sortie :
    - Message de confirmation du statut de l’API
  + Exemple :
    - curl.exe -X GET -H "Authorization: Basic ABCDEF123456" http://127.0.0.1:8080/secapichk
* Route pour la prédiction de la classe d’une image
  + Route :
    - <http://127.0.0.1:8080/predict/> (POST)
  + Paramètre(s) :
    - Nom de connexion de l’utilisateur
    - Mot de passe
    - Chemin d’une image
  + Sortie :
    - Prédiction
    - Précision de la prédiction (Accuracy)
    - Durée de la prédiction
  + Exemple :
    - curl.exe -X POST -F "image=@image\_folder/image\_to\_predict.jpeg" -H "Authorization: Basic ABCDEF123456" http://127.0.0.1:8080/predict/
* Route de lancement d’un nouvel entraînement (accessible aux administrateurs uniquement)
  + Route :
    - pour l’entraînement complet : <http://127.0.0.1:8080/train/full/> (POST)
    - pour l’entraînement « démo » : <http://127.0.0.1:8080/train/mini/> (POST)
  + Paramètre(s) :
    - Nom de connexion de l’utilisateur
    - Mot de passe
  + Sortie :
    - Message de fin d’entraînement avec confirmation du chemin vers le nouveau modèle
    - Message d’erreur si l’utilisateur n’est pas « admin »
  + Exemple :
    - Entraînement complet : curl.exe -X POST -H "Authorization: Basic ABCDEF123456" <http://127.0.0.1:8080/train/full/>
    - Entraînement « démo » : curl.exe -X POST -H "Authorization: Basic ABCDEF123456" http://127.0.0.1:8080/train/mini/
* Route pour ajout d’une nouvelle image
  + Route :
    - <http://127.0.0.1:8080/addpict/> (POST)
  + Paramètre(s) :
    - Nom de connexion de l’utilisateur
    - Mot de passe
    - Chemin d’une image
    - Type de plante
    - Type de maladie
  + Sortie :
    - Message de confirmation de l’ajout de la nouvelle image pour vérification
  + Exemple :
    - curl.exe -X POST -F "plante=Apple" -F "maladie=healthy" -F "image=@image\_folder/image\_to\_add.jpeg" -H "Authorization: Basic ABCDEF123456" http://127.0.0.1:8080/addpict/
* Route pour qu’un utilisateur accède à son historique (logs)
  + Route :
    - <http://127.0.0.1:8080/gimmelogs> (GET)
  + Paramètre(s) :
    - Nom de connexion de l’utilisateur
    - Mot de passe
  + Sortie :
    - Eléments de la log liés à l’utilisateur connecté sous format JSON
  + Exemple :
    - curl.exe -X GET -H "Authorization: Basic ABCDEF123456" http://127.0.0.1:8080/gimmelogs
* Route pour voir l’entièreté des logs (accessible aux administrateurs uniquement)
  + Route :
    - <http://127.0.0.1:8080/getalllogs> (GET)
  + Paramètre(s) :
    - Nom de connexion de l’utilisateur
    - Mot de passe
  + Sortie :
    - Entièreté des logs sous format JSON
    - Message d’erreur si l’utilisateur n’est pas « admin »
  + Exemple :
    - curl.exe -X GET -H "Authorization: Basic ABCDEF123456" http://127.0.0.1:8080/getalllogs

## Testing & Monitoring

### Les tests unitaires

Différents tests unitaires sont à mettre en place afin de vérifier le bon fonctionnement de différents processus.

Les tests unitaires pour valider le bon fonctionnement du modèle lors de l’entraînement :

* Test d’existence des fichiers nécessaires (CSV principal « New Plant Disease », Base d’images, Base de Logs)
* Test du bon fonctionnement des commandes d’entraînement (sur très peu de photos)
* Vérification de la fin de l’entraînement

Les tests unitaires pour valider le bon fonctionnement du modèle lors de la prédiction :

* Test d’existence des fichiers nécessaires (modèle EfficentNetV2S, Base de Logs)
* Test sur prédiction d’une image connue du scope de validation

Les tests unitaires pour valider le bon fonctionnement des différentes routes de l’API :

* Test route de statut
* Test route d’authentification (avec utilisateurs test spécifique)
* Test route de prédiction connue
* Test route d’un ajout de photo test
* Test route d’accès à l’historique (admin ou utilisateur)

Les tests unitaires pour valider le bon fonctionnement du process d’ajout de nouvelles données :

* Test d’existence des fichiers nécessaires (CSV principal « New Plant Disease », Base tampon d’ajouts, Base de Logs)
* Vérification de l’existence de la nouvelle photo dans le bon répertoire + dans la base tampon

Les tests unitaires pour valider le bon fonctionnement du process d’ingestion de nouvelles données :

* Test d’existence des fichiers nécessaires (CSV principal « New Plant Disease », Base tampon d’ajouts, Base de Logs)
* Vérification de l’existence de la nouvelle photo dans le bon répertoire + dans la base principale « New Plant Diseases »

ATTENTION : Il a été décidé que le processus d’ajout final dans la base principale se fera après vérification par un humain de la cohérence des nouvelles photos et de la proposition de classification fournie.

### Le monitoring du modèle

Une fois le modèle mis en place, il faut le monitorer et prendre des décisions en fonction de cette surveillance.

A un instant donné, la performance du modèle sera évalué en fonction de :

* L’accuracy de la prédiction
* Du temps de réponse de la prédiction
* De la satisfaction client sur la prédiction
* De la satisfaction client sur le temps de réponse

Le modèle devra être ré-entrainer :

* Lorsqu’au moins 20% des scores d’accuracy sont trop faibles sur une semaine (en dessous de 95% accuracy)
* Lorsqu’au moins 20% des notes utilisateurs sont mauvaises sur une semaine (note de moins de 3 sur 5)
* Si le temps de réponse est multiplié par 2 par rapport au temps de référence

Le modèle sera alors ré-entrainer sur l’intégralité du jeu de données avec les nouvelles données ajoutées potentiellement.

Si le modèle n’atteint pas le seuil de performance requis, un mail d’alerte sera envoyé aux administrateurs de l’application pour analyse et action.

## Schéma d’implémentation

Vous trouverez présenté, ci-dessous, le schéma d’implémentation de la solution que nous développons afin d’aider les agriculteurs dans la reconnaissance de leurs cultures ainsi que des potentielles maladies associées :

