



|  |
| --- |
| UNIVERSITE MOULAY ISMAIL –UMI-  BACHELOR EN GENIE INFORMATIQUE -BGI- |
| Computer Vision |
| janvier 2  Réaliser par : Khalid Naimi  Encadré par : Mr Ayoub  Année Universitaire : 2020-2021 |

Soutenu devant le jury :

Mr Bekri Ali, Professeur à la Faculté des Sciences- Meknès

Mr Benhlima Said, Professeur à la Faculté des Sciences- Meknès

Mr Oubelkacem Ali, Professeur à la Faculté des Sciences- Meknès

# Remerciements :

Avant tout développement sur cette expérience, il apparait opportunité de commencer ce rapport par des remerciements à ceux qui nous ont beaucoup appris au cours de ce projet.

Nous tenons à remercier dans un premier temps, toute l’équipe pédagogique de Notre Institut et les intervenants professionnels responsables de la formation génie informatique.

Nous exprimons toute notre gratitude et nos remerciements à Monsieur Ali BEKRI, Monsieur OUBELKACEM qui n’ont pas cessé de nous encourager pendant la durée du projet. Nous les remercions également pour l’intérêt qu’ils ont porté à notre travail, pour leurs encouragements, et pour leurs conseils.

Nous tenons à remercier également, Monsieur BENHLIMA membre de jury.

Sans oublier Monsieur Ayoub qui nous a fait apprendre les bases du machine learning .

Finalement, nous ne manquerons pas d’adresser nos remerciements à l’ensemble de nos collègues pour leurs efforts pendant toutes les années précédentes.

**Table of Contents:**

[Remerciements : 2](#_Toc60525478)

[Introduction : 5](#_Toc60525479)

[Chapitre 1 : contexte général du projet 6](#_Toc60525480)

[Vision du projet : 6](#_Toc60525481)

[Architecture du projet : 7](#_Toc60525482)

[Partie 1 : 7](#_Toc60525483)

[Partie 2 : 7](#_Toc60525484)

[Chapitre 2 : Bibliothèques et outils 8](#_Toc60525485)

[I. Outils : 8](#_Toc60525486)

[1. Jupyter : 8](#_Toc60525487)

[2. Google colab : 9](#_Toc60525488)

[II. Bibliothèques : 9](#_Toc60525489)

[1. Matplotlib : 9](#_Toc60525490)

[2. NumPy : 10](#_Toc60525491)

[3. Keras : 10](#_Toc60525492)

[4. Scikit-learn : 11](#_Toc60525493)

[5. OpenCv : 11](#_Toc60525494)

[6. Tkinter : 12](#_Toc60525495)

[Chapitre 3 : Mise en œuvre de projet 12](#_Toc60525496)

[Partie 1 : Bag of features 12](#_Toc60525497)

[Les importations des bibliothèques : 12](#_Toc60525498)

[Importation des images 13](#_Toc60525499)

[Transformer le nom des classes en id : 14](#_Toc60525500)

[Détection des features de chaque image : 15](#_Toc60525501)

[Nettoyage des listes : 17](#_Toc60525502)

[Obtention de x et y : 18](#_Toc60525503)

[L’entrainement du modèle : 19](#_Toc60525504)

[Le test du modèle : 19](#_Toc60525505)

[Discussion : 20](#_Toc60525506)

[Partie 2 : VGG-16 21](#_Toc60525507)

[Les importations : 21](#_Toc60525508)

[Importation du VGG-16 de keras: 21](#_Toc60525509)

[Création du modèle : 21](#_Toc60525510)

[Data/Test Generator : 22](#_Toc60525511)

[Entrainement du modèle (couche ajouter) : 23](#_Toc60525512)

[Représentation graphique des résultats : 24](#_Toc60525513)

[La méthode recogniseImage(image\_path) 25](#_Toc60525514)

[Discussion : 25](#_Toc60525515)

[Partie 3 : Interface graphique 26](#_Toc60525516)

[Création de la fenêtre : 26](#_Toc60525517)

[Création des champs à remplir : 26](#_Toc60525518)

[Radio button Vgg16 / features : 27](#_Toc60525519)

[Création des boutons : 28](#_Toc60525520)

[Lancement de l’application : 28](#_Toc60525521)

[Conclusion : 29](#_Toc60525522)

[Références : 31](#_Toc60525523)

# Introduction :

Le programme qu’on va vous présenter consiste, brièvement, en partie 1 à entrainer un classificateur d’images des feuilles d’arbres, et en partie 2 à adapter le modèle VGG16

A notre classification.

Lors de ce projet, on va particulièrement utiliser le langage de programmation Python.

Plusieurs outils et bibliothèques ont été utilisés permettant le développement des différents modules réalisés au cours du projet.

Le résultat de ce projet est un modèle capable de classifier correctement les images selon leurs catégories.

Ce rapport se décompose en quatre parties :

* Contexte général du projet : comprenant la présentation du cahier de charges, la description du sujet et sa problématique.
* Bibliothèques et outils : comprenant les outils utilisés pour l’entrainement du modèle.
* Mise en œuvre de projet : comprenant les étapes détaillées suivies au cours du projet.

# Chapitre 1 : contexte général du projet

Le projet proposé s’intéresse au suivi de l’état sanitaire des plantes en utilisant le traitement d’image et les techniques d’apprentissage profond (Deep Learning). Une application doit être développée dont l’objectif est de donner des informations sur l’état sanitaire des plantes à partir des images foliaires. Ces informations seront utilisées pour la prise des décisions sur le choix du traitement des cultures touchées.

## **Vision du projet :**

Le but de ce projet est d’entrainer un algorithme d’apprentissage supervisé sur les bag-of-features extraites des images, afin de créer un classificateur capable de prédire la classe des images de feuilles d’arbres.

Puis on va apprendre à utiliser la bibliothèque Keras du deep learning , afin de pouvoir utiliser le Vgg16 pour faire la recognition des classes des feuilles d’arbres .

Enfin on est sensé de comparer les résultats de la partie une et deux.

## **Architecture du projet :**

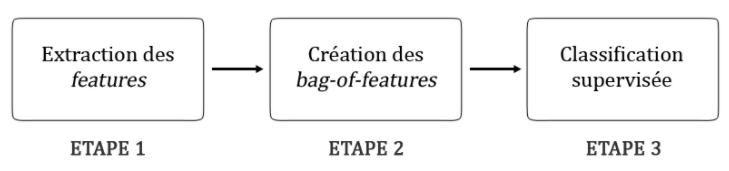
Le problème de classification d'images est posé formellement de la manière suivante :

* Il y a 39 classes d'images possibles. L'ensemble {0,1,...,K−1} définit les labels des différentes classes (exemple : 0 = " Apple\_\_\_Apple\_scab" et 1 = " Apple\_\_\_Black\_rot")
* Nous avons une collection de N images en entrée : {Xi} avec i∈{1,...,N}
* Les classes des N images sont connues à l'avance : chaque image Xi est étiquetée par yi∈{0,1,...,K−1}

Le but est de classifier correctement les nouvelles images (dossier « validation » ) avec nos deux méthodes :

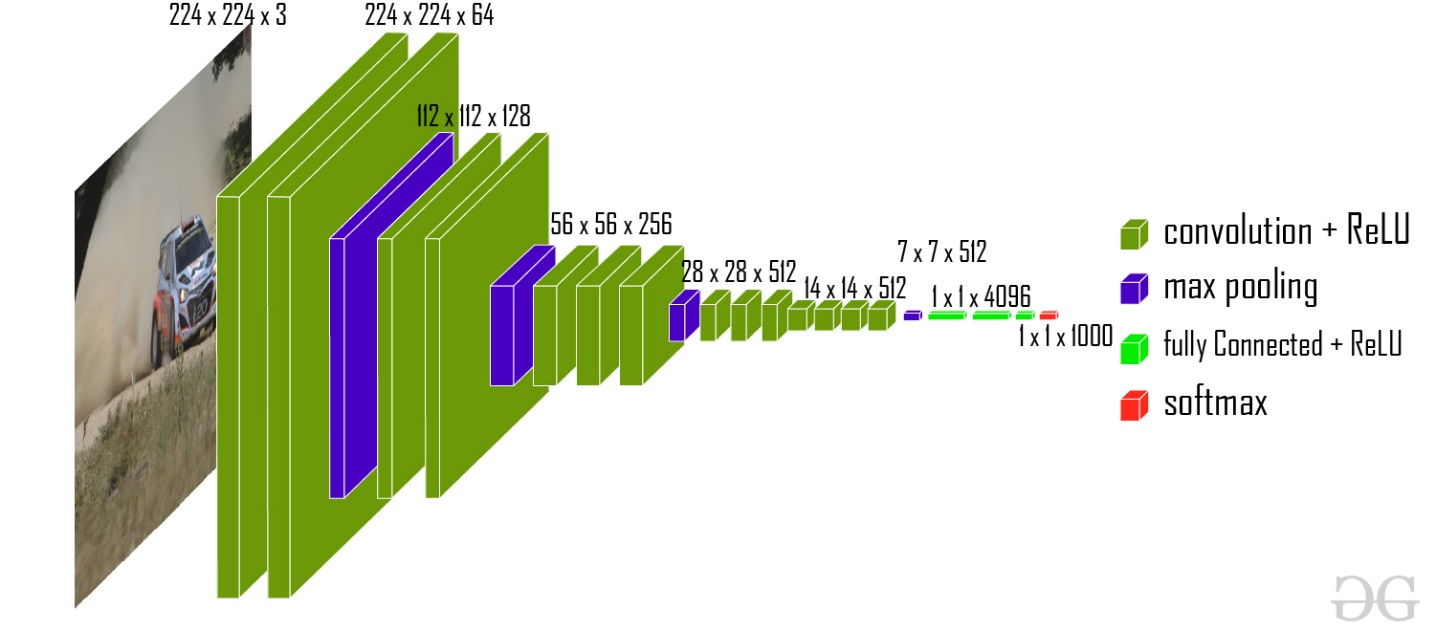
### Partie 1 :

Le schéma ci-dessous illustre la méthode 1 utilisée pour résoudre ce problème :



### Partie 2 :

Dans cette partie, nous allons utiliser Keras, une bibliothèque très intuitive de Deep Learning en Python.



Architecture Vgg 16.

# Chapitre 2 : Bibliothèques et outils

## **Outils :**

### Jupyter :



Project Jupyter est une organisation à but non lucratif crée pour "développer des logiciels open source, des standards ouverts et des services pour l'informatique interactive dans des dizaines de langages de programmation". Lancé d'IPython en 2014 par Fernando Pérez, Project Jupyter prend en charge les environnements d'exécution dans plusieurs dizaines de langues . Le nom du projet Jupyter est une référence aux trois principaux langages de programmation pris en charge par Jupyter, à savoir Julia, Python et R, ainsi qu'un hommage aux cahiers de Galileo enregistrant la découverte des lunes de Jupiter.

### Google colab :



Google Colaboratory, parfois appelé Colaboratory en abrégé, est un service cloud de Google qui réplique Jupyter Notebook dans le cloud. Vous n'avez rien à installer sur votre système pour l'utiliser. Dans la plupart des cas, vous utilisez Colaboratory comme vous le feriez pour une installation de bureau de Jupyter Notebook.

## **Bibliothèques :**

### Matplotlib :



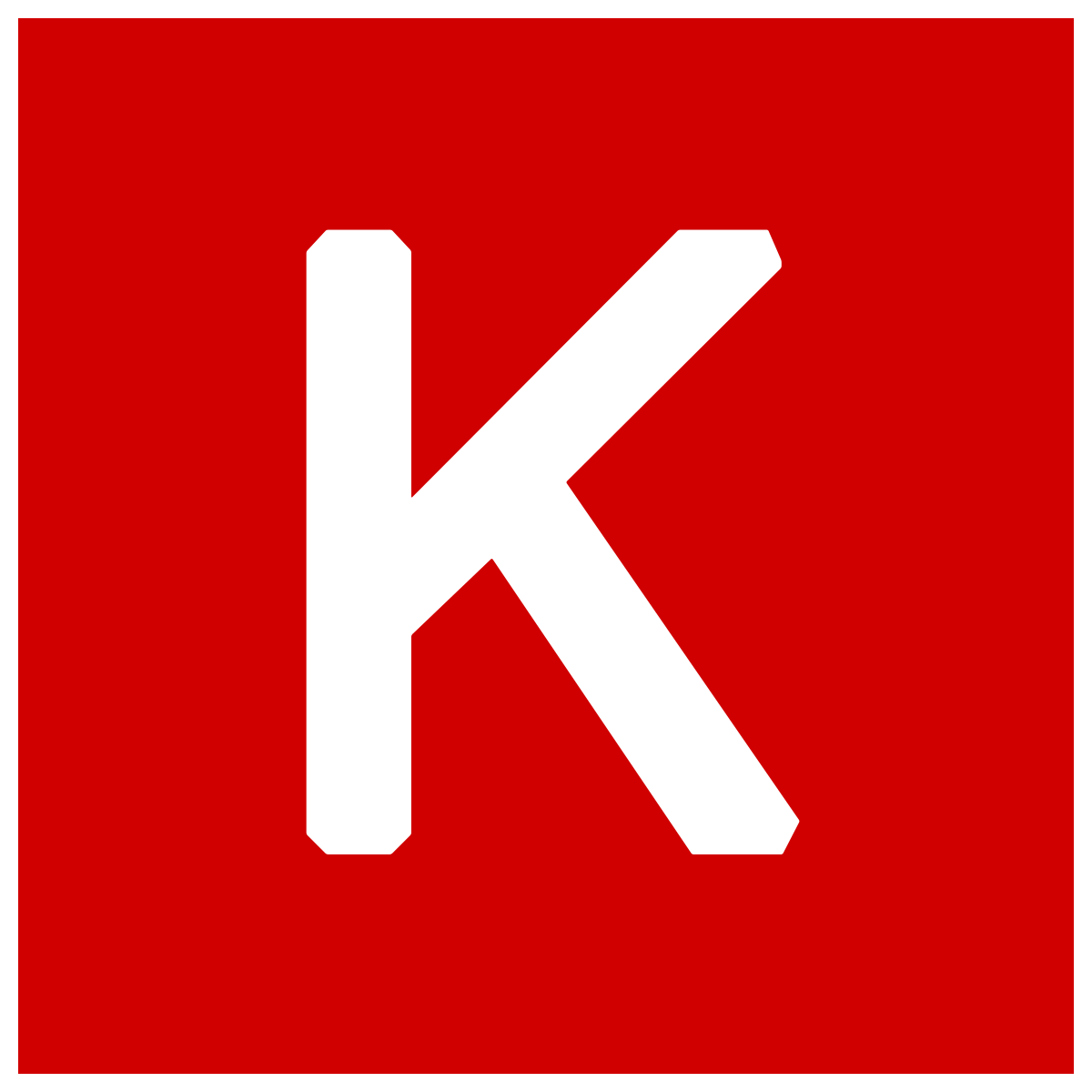
Matplotlib est une bibliothèque de traçage pour le langage de programmation Python et son extension de mathématiques numériques NumPy. Il fournit une API orientée objet pour incorporer des tracés dans des applications à l'aide de boîtes à outils GUI à usage général telles que Tkinter, wxPython, Qt ou GTK +. Il existe également une interface procédurale «pylab» basée sur une machine à états (comme OpenGL), conçue pour ressembler étroitement à celle de MATLAB, bien que son utilisation soit déconseillée. Scie utilise Matplotlib.

### NumPy :



NumPy (NUM-pee)) est une bibliothèque pour le langage de programmation Python, ajoutant la prise en charge de grands tableaux et matrices multidimensionnels, ainsi qu'une grande collection de fonctions mathématiques de haut niveau pour opérer sur ces tableaux.L'ancêtre de NumPy, Numeric, a été créé à l'origine par Jim Hugunin avec les contributions de plusieurs autres développeurs. En 2005, Travis Oliphant a créé NumPy en incorporant des fonctionnalités du Numarray concurrent dans Numeric, avec des modifications importantes. NumPy est un logiciel open source et compte de nombreux contributeurs.

### Keras :



Keras est une bibliothèque de logiciels open source qui fournit une interface Python pour les réseaux de neurones artificiels. Keras agit comme une interface pour la bibliothèque TensorFlow.

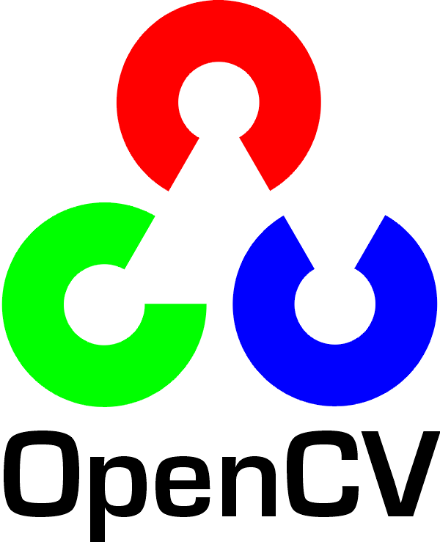
Conçu pour permettre une expérimentation rapide avec des réseaux de neurones profonds, il se concentre sur être convivial, modulaire et extensible. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-end Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur de Google. Chollet est également l'auteur du modèle de réseau neuronal profond XCeption.

### Scikit-learn :



Scikit-learn (anciennement scikits.learn et également connu sous le nom de sklearn) est une bibliothèque d'apprentissage automatique de logiciels gratuits pour le langage de programmation Python.Il propose divers algorithmes de classification, de régression et de clustering, y compris les machines vectorielles de support, les forêts aléatoires, l'augmentation de gradient, k-means et DBSCAN, et est conçu pour interagir avec les bibliothèques numériques et scientifiques Python NumPy et SciPy.

### OpenCv :



OpenCV (Open Source Computer Vision Library) est une bibliothèque de fonctions de programmation principalement destinées à la vision par ordinateur en temps réel. Développé à l'origine par Intel, il a ensuite été soutenu par Willow Garage puis Itseez (qui a ensuite été acquis par Intel). La bibliothèque est multiplateforme et gratuite pour une utilisation sous la licence open-source Apache 2. À partir de 2011, OpenCV propose une accélération GPU pour les opérations en temps réel.

### Tkinter :



Tkinter (de l'anglais Tool kit interface) est la [bibliothèque graphique](https://fr.wikipedia.org/wiki/Biblioth%C3%A8que_graphique) [libre](https://fr.wikipedia.org/wiki/Logiciel_libre) d'origine pour le langage [Python](https://fr.wikipedia.org/wiki/Python_(langage)), permettant la création d'[interfaces graphiques](https://fr.wikipedia.org/wiki/Interface_graphique). Elle vient d'une adaptation de la bibliothèque graphique [Tk](https://fr.wikipedia.org/wiki/Tk_(informatique)" \o "Tk (informatique)) écrite pour [Tcl](https://fr.wikipedia.org/wiki/Tool_Command_Language).

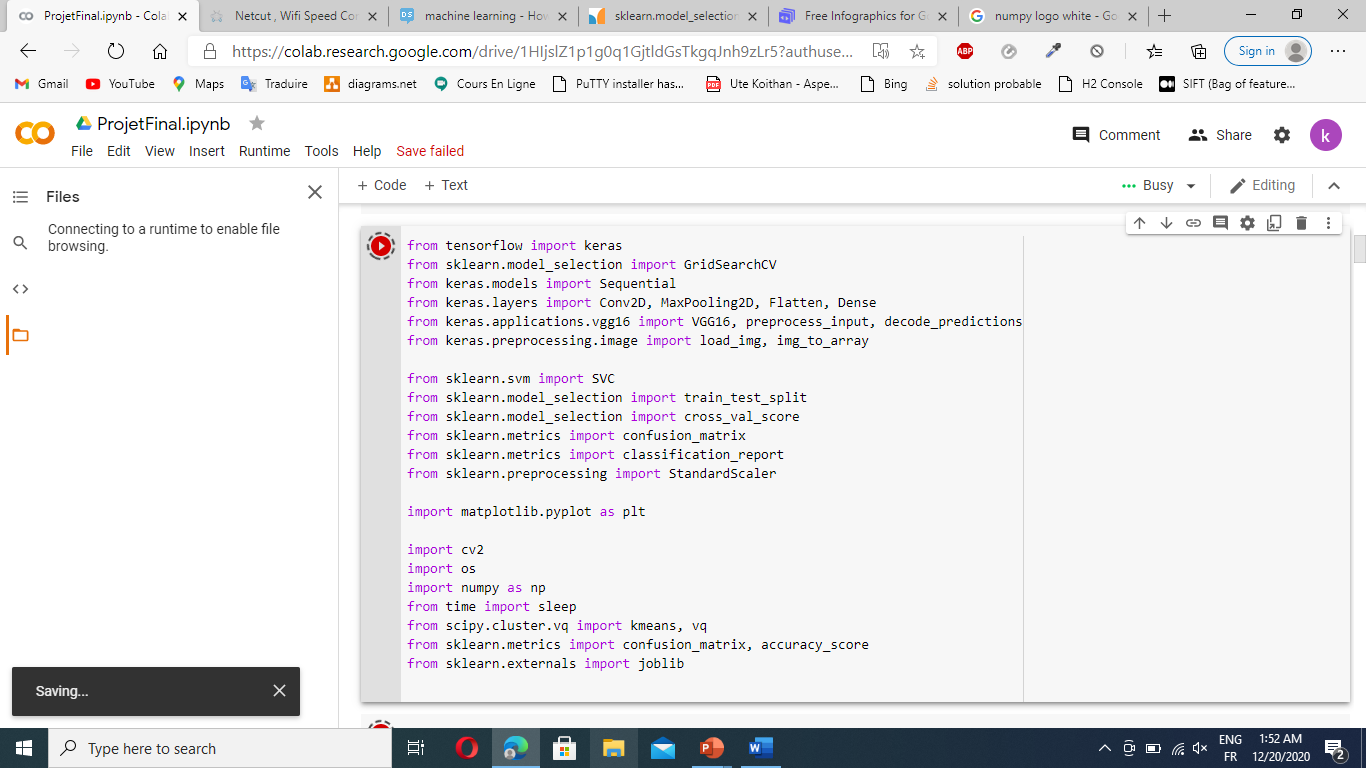
# Chapitre 3 : Mise en œuvre de projet

## **Partie 1 : Bag of features**

On a exécuté le projet en suivant les étapes suivantes :

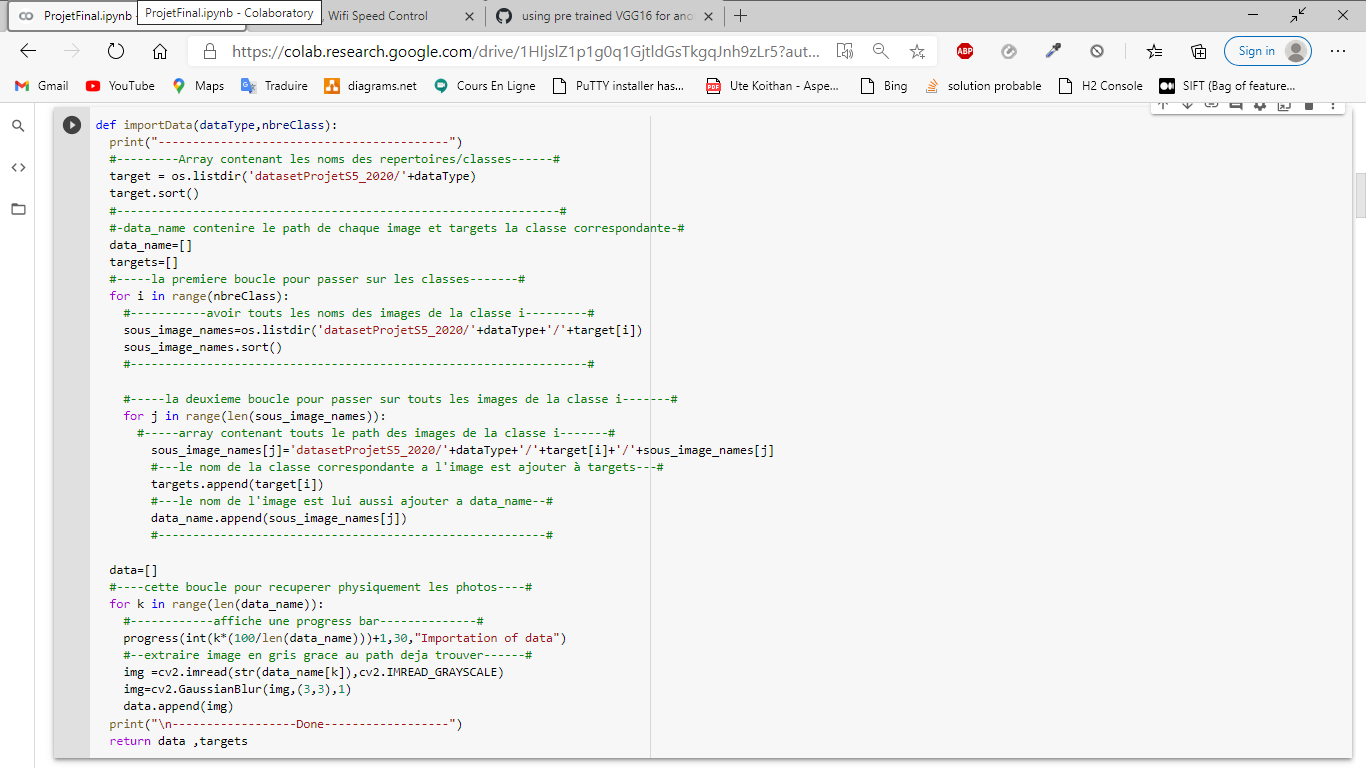
### Les importations des bibliothèques :

On importe les bibliothèques :



### Importation des images

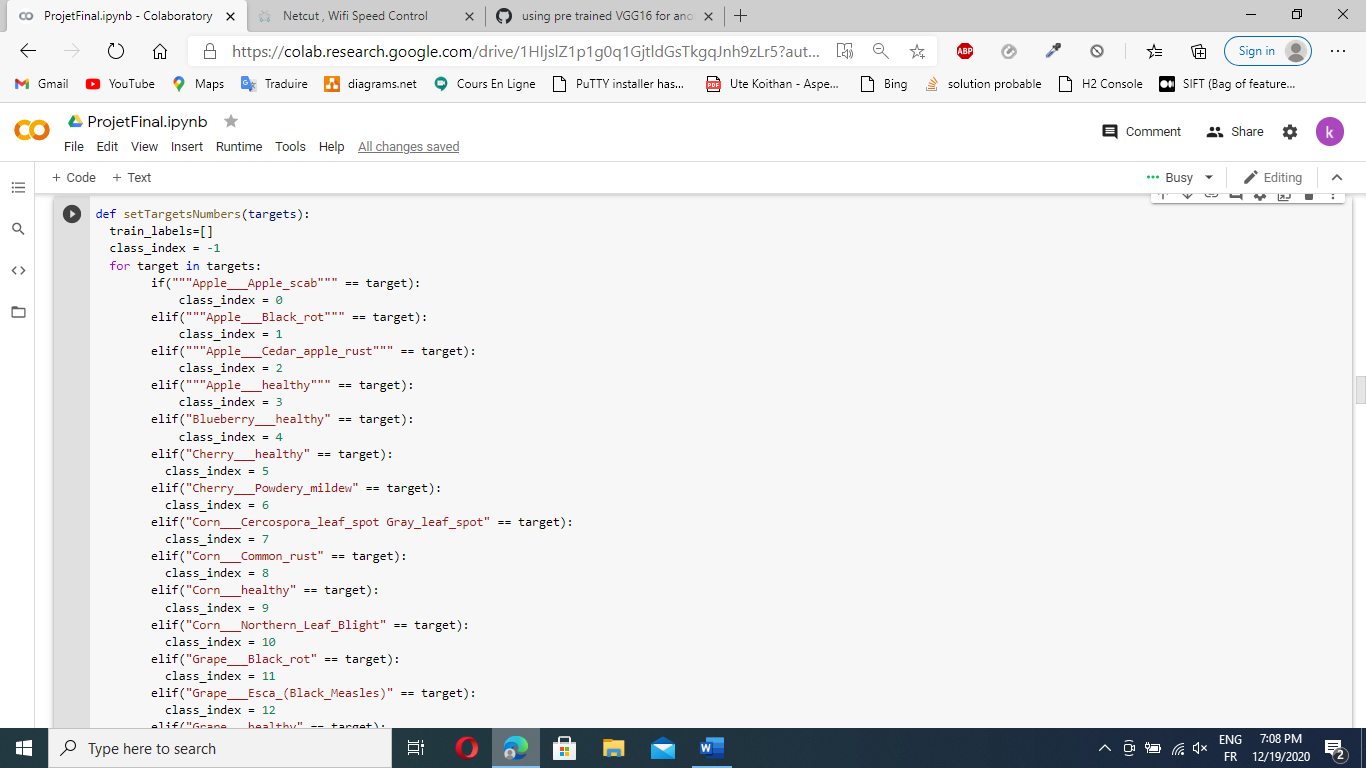
Grace a cette méthode ( importData() ) en importe les images. La fonction prend en paramètre le dataType, c’est-à-dire le type des images qu’on veut (train ou test). Et le nbreClass indique le nombre de classes avec lequel on veut travailler (39 ou moins).



La méthode retourne les images et leurs labels.

### Transformer le nom des classes en id :

La méthode simple setTargetsNumbers permet de faire ce travail.



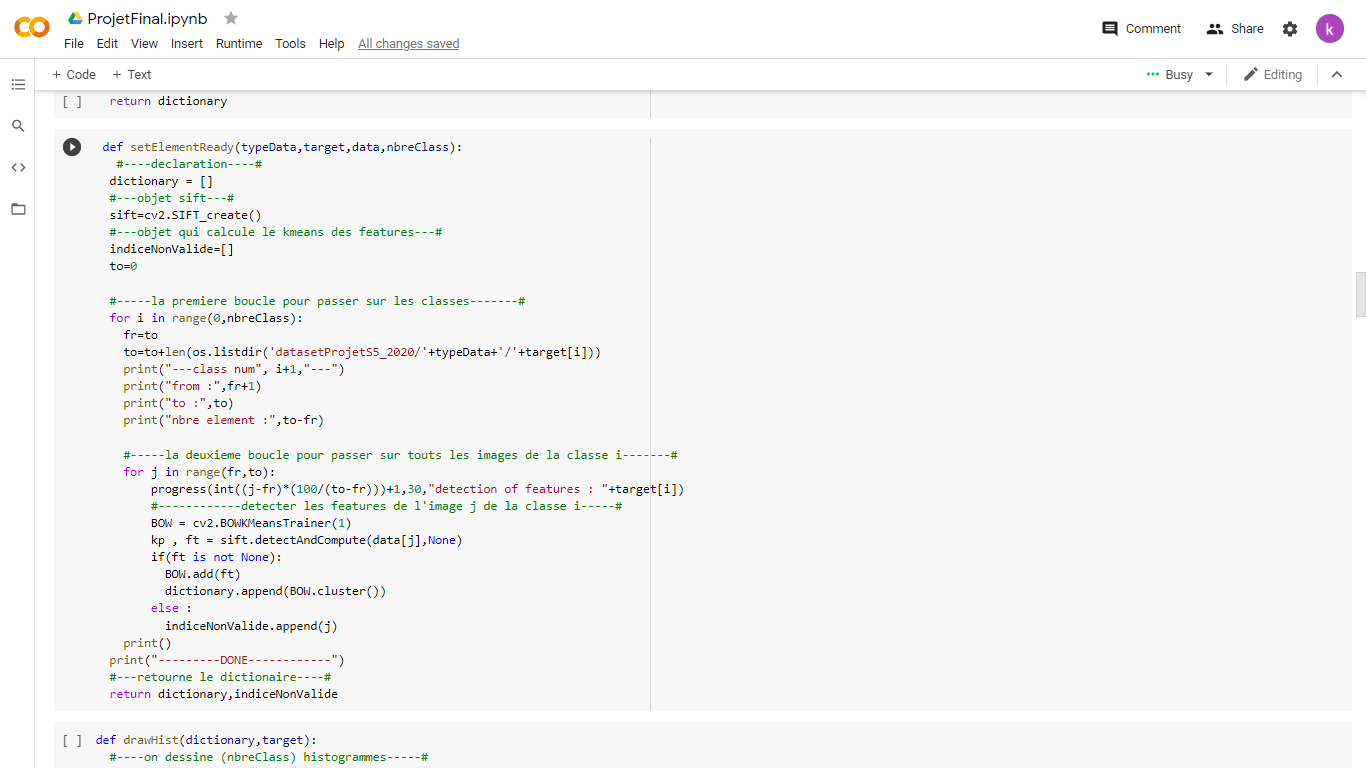
La méthode retourne la nouvelle list (y\_train).

### Détection des features de chaque image :

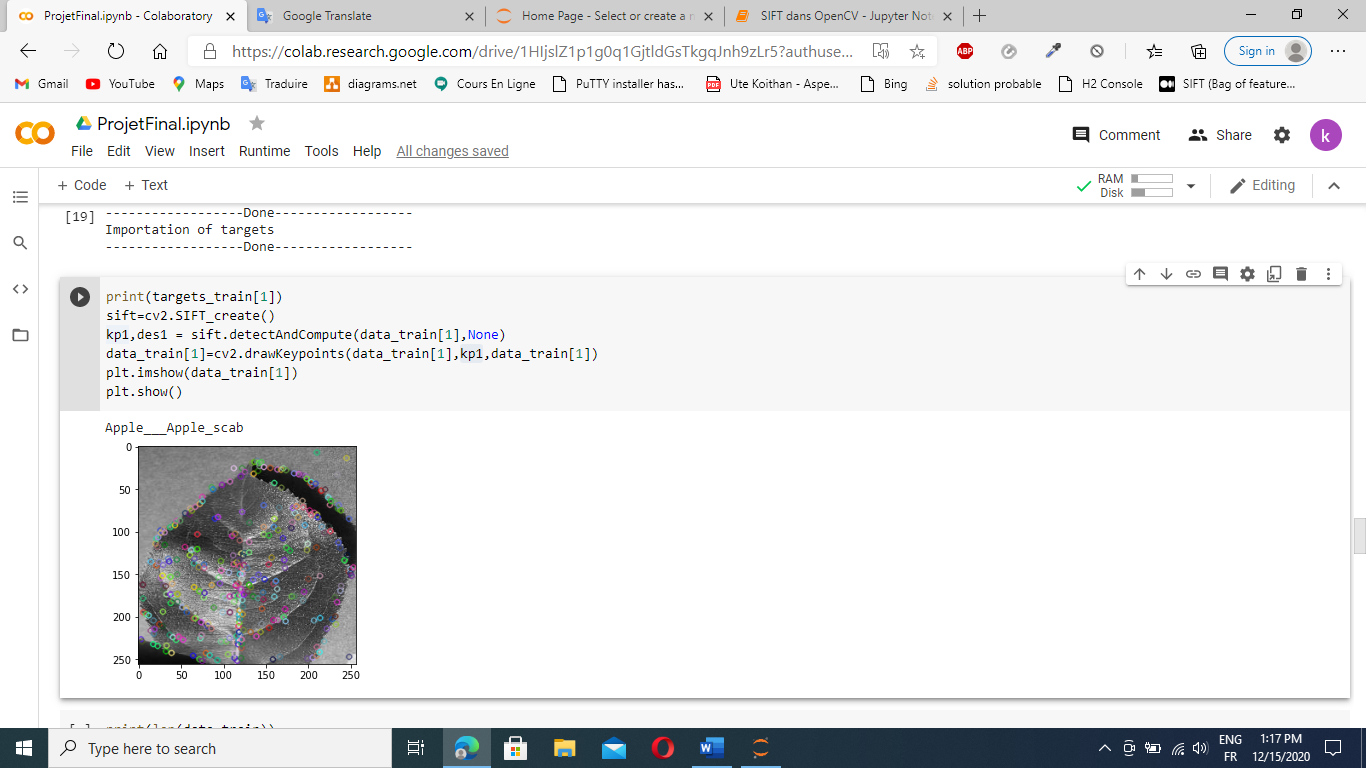
La méthode setElementReady a trois paramètres :

* typeData : il faut préciser si les éléments et pour entrainement ou test.
* target : les labels des images.
* Data : array des images.
* nbreClass : le nombre de classe avec lequel on veut travailler

Cette méthode détecte les features de chaque image et les rassemble dans une liste.



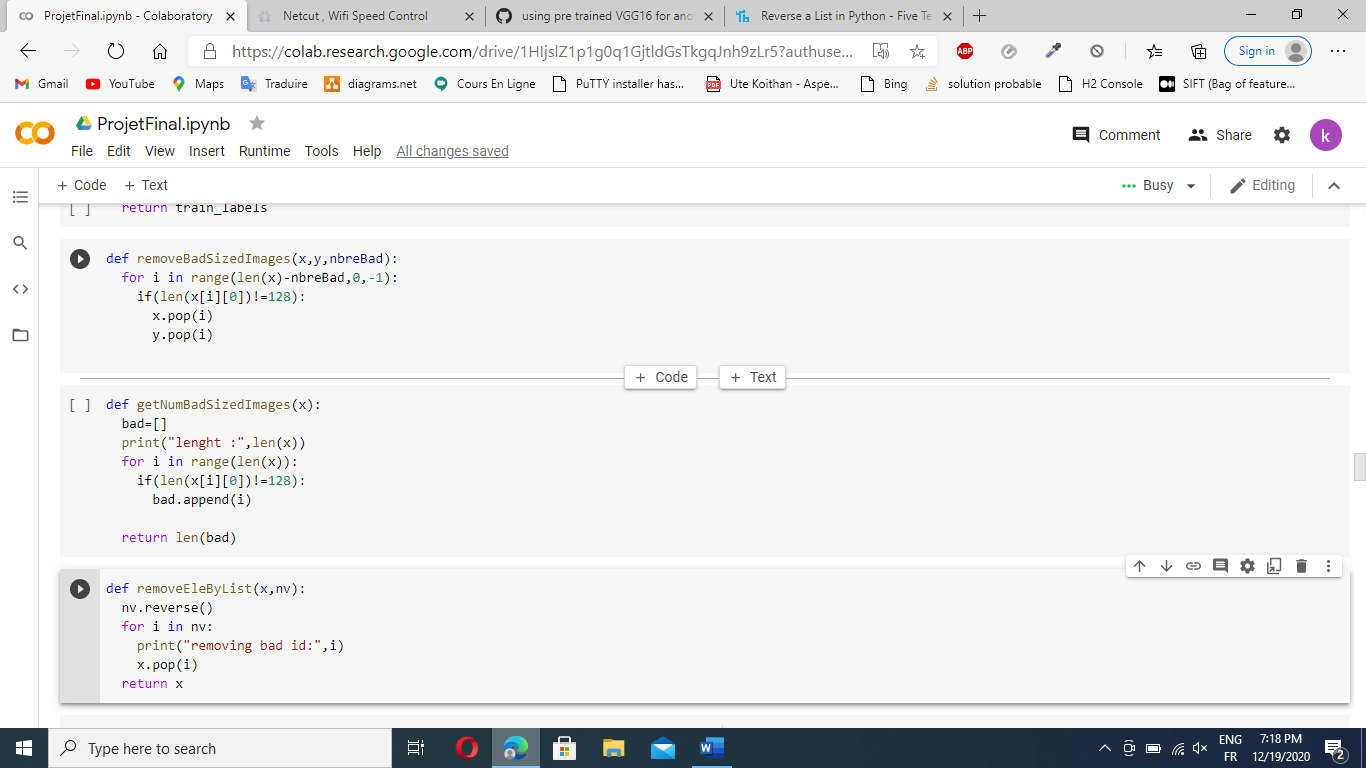
La méthode retourne la liste des features (x\_train) et les indices des éléments dont elle a échoué d’extraire les features.



Exemple des features d’une image

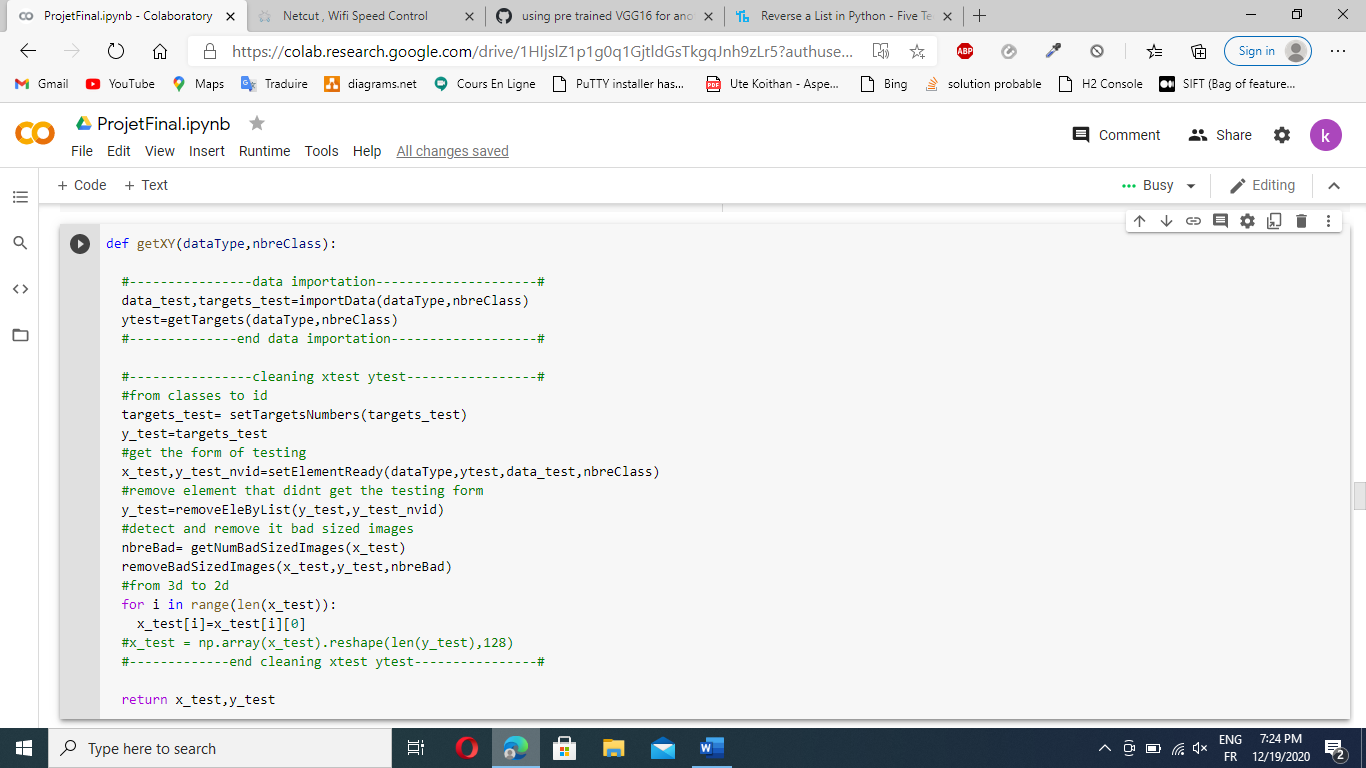
### Nettoyage des listes :

Ces 3 méthodes sont utilisées pour nettoyer nos données qu’on va utiliser en entrainement ou en test.



### Obtention de x et y :

Cette méthode getXY rassemble toutes les méthodes déjà décrites en haut, elle permet de nous donner nos données qu’on va utiliser directement en entrainement ou test, ci-dessous le corps de la fonction.

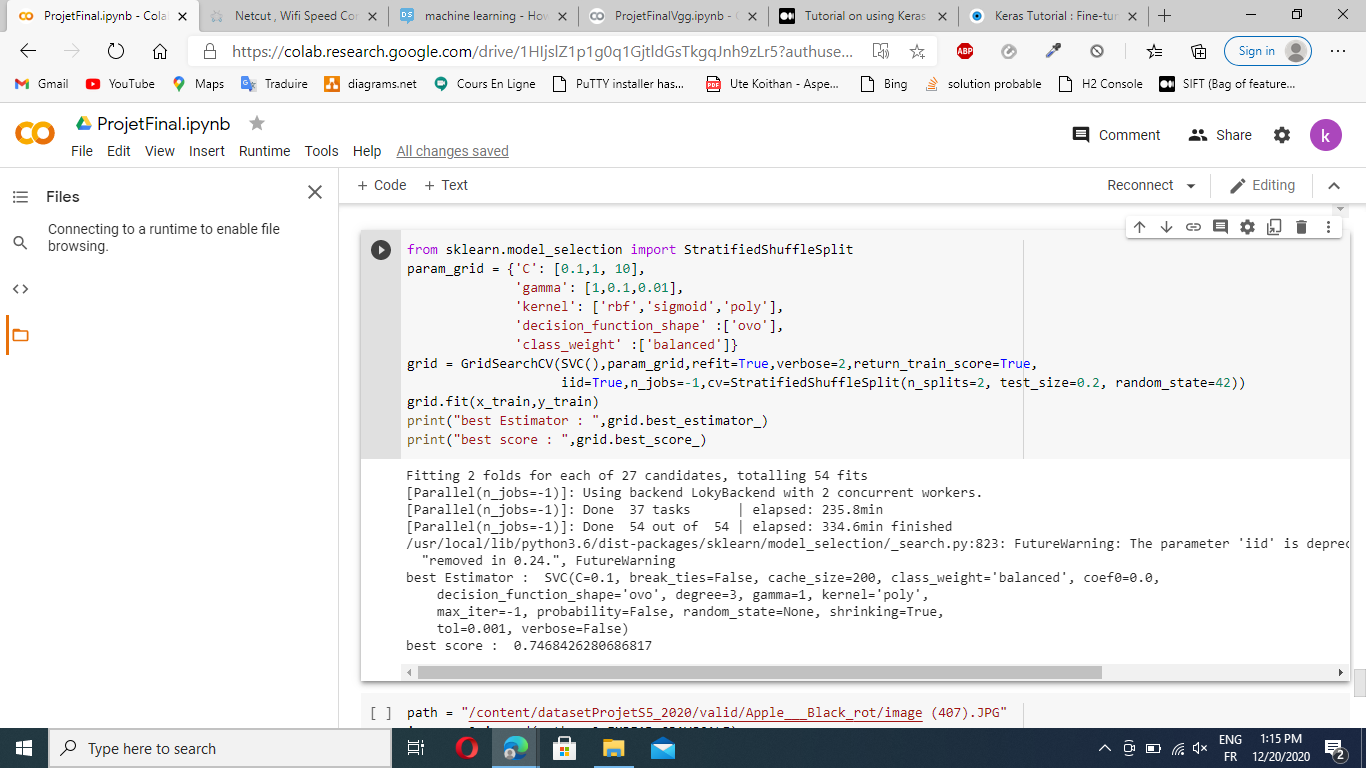


La méthode retourne soit x\_train et y\_train , soit x\_test et y\_test . Celui-ci dépend du paramètre dataType (‘train’,’test’).

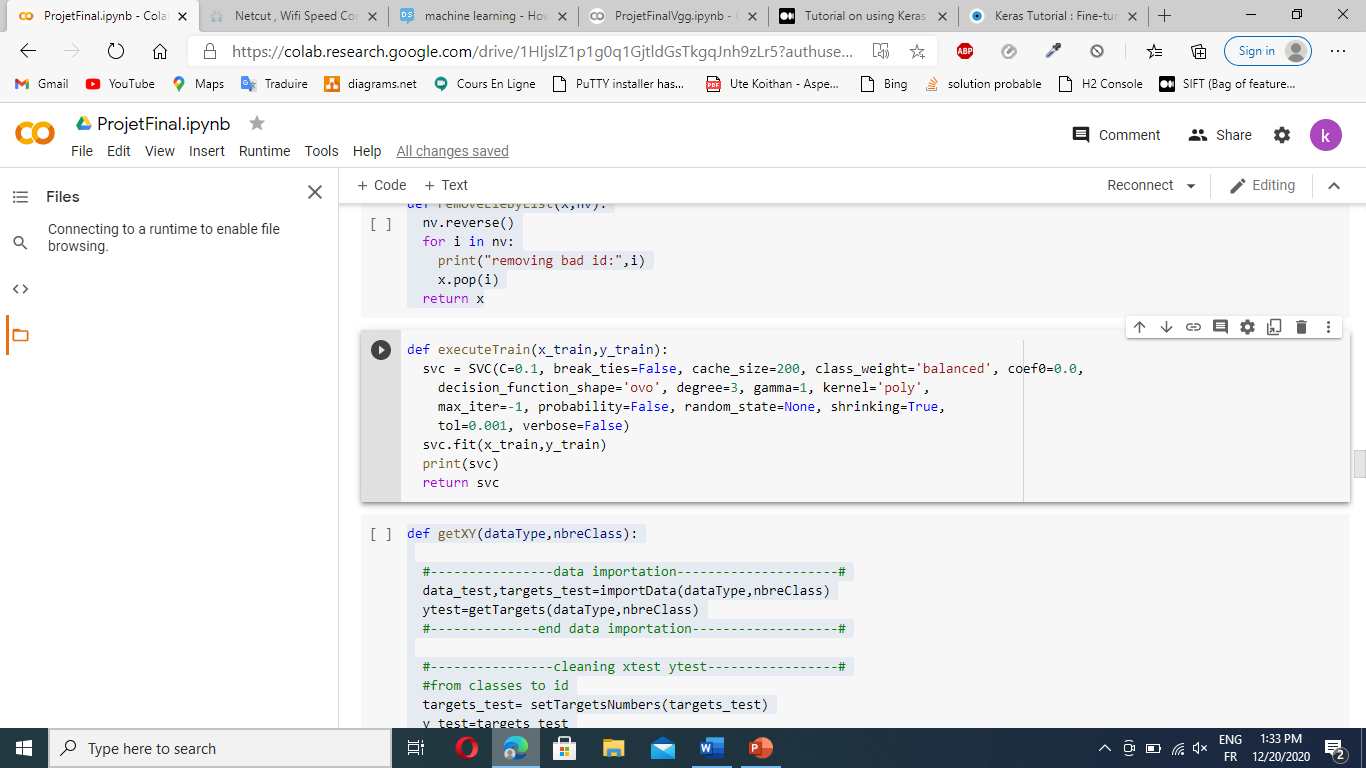
### L’entrainement du modèle :

Pour cela j’ai utilisé la méthode simple executeTrain qui prend en arguments le x\_train et y\_train.

Mais premièrement pour bien choisir les paramètres de mon modèle svm , j’ai utilisé GridSearchCV.



Maintenant puisque j’ai les paramètres optimaux je peux entrainer mon modèle :



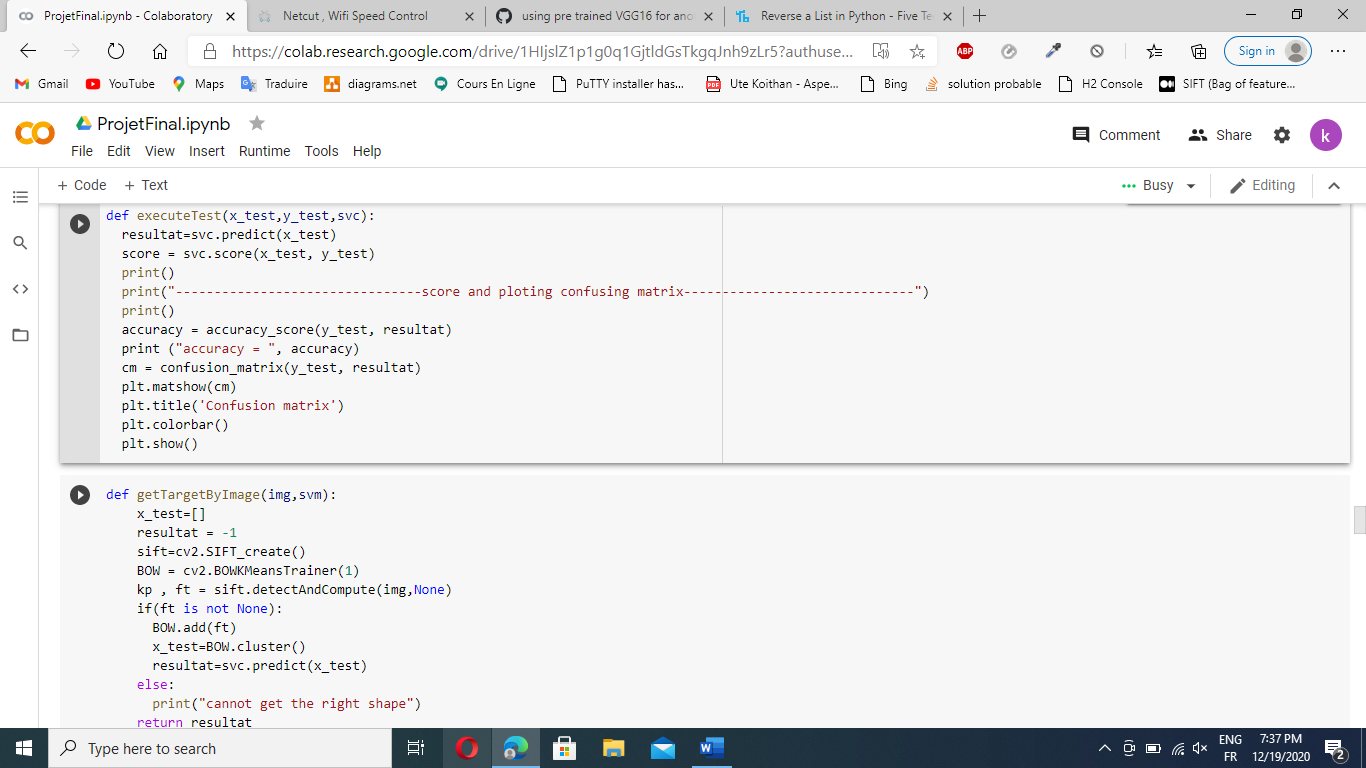
La méthode retourne le modèle.

### Le test du modèle :

Pour cela j’ai utilisé deux méthodes :

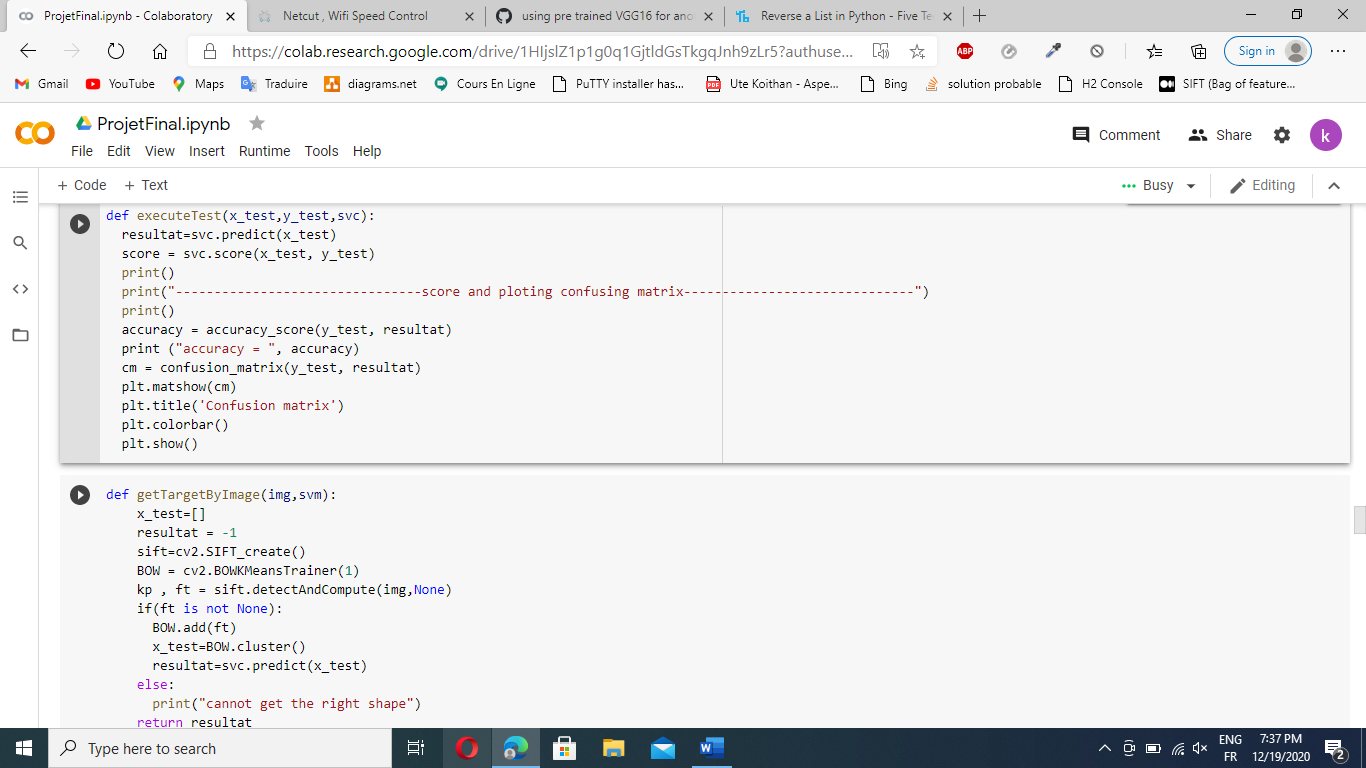
* La première fonction executeTest :

Cette fonction prend en argument x\_test , y\_test et le modèle et elle permet de calculer le score et de tracer la matrice de confusion pour mieux présenter les résultats



* La deuxième fonction getTragetByImage :

Cette dernière prend en argument une image et le modèle, puis elle affiche la prédiction.



### Discussion :

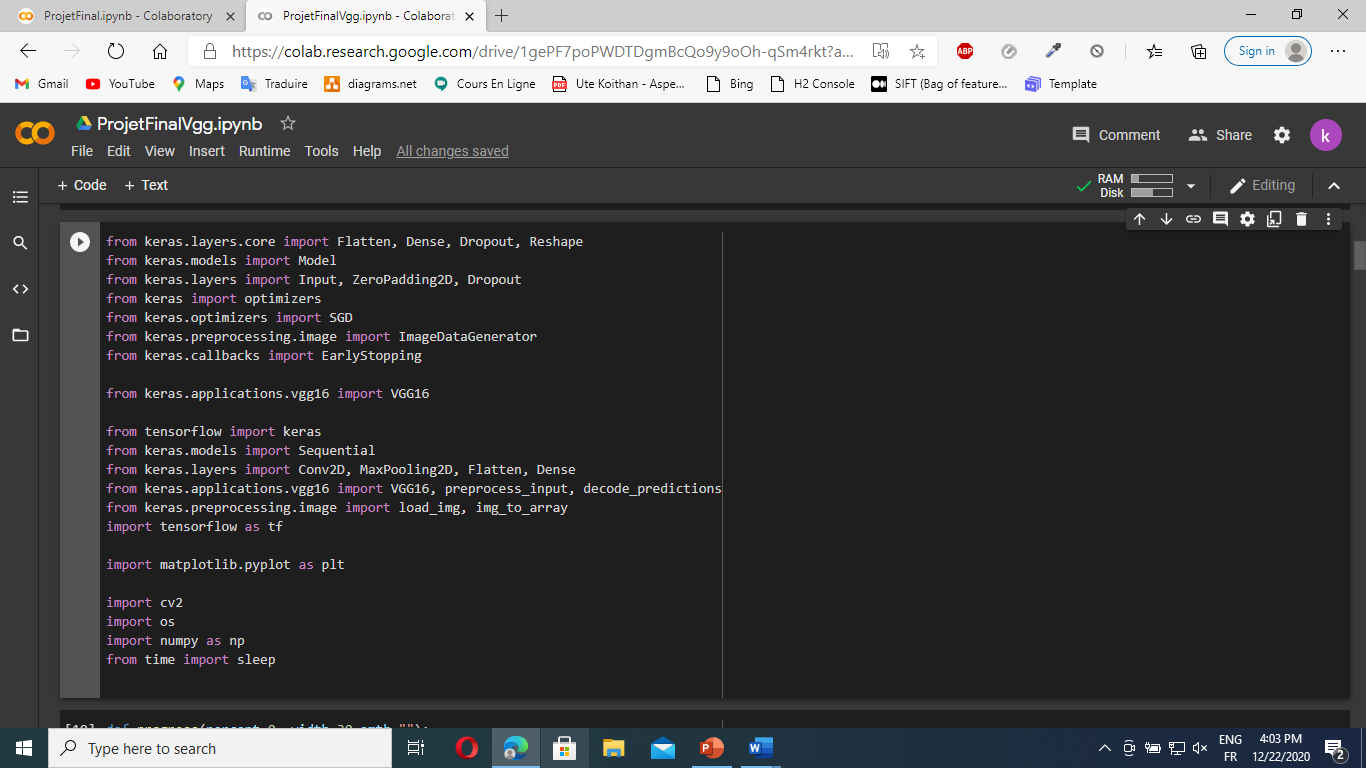
* Première remarque évidente c’est le score, on remarque que le score n’était pas très bon, 74% n’est pas très précis.
* Cette méthode ne consomme pas trop de ressources, si on la compare aux méthodes de deep learning.
* L’entrainement du modèle dans cette méthode est relativement rapide.
* Afin de l’améliorer, on peut appliquer autres filtres pour mieux détecter les features des images.
* On peut également utiliser plus de combinaison de paramètres dans la méthode gridSearchCV , mais ça consomme beaucoup de ressources et de temps.
* Pour mon noyau ‘poly’ je peux optimiser le score en augmentant le degré , mais ca risque de générer un modèle surentrainer.

## **Partie 2 : VGG-16**

Lors de la partie deux, j’ai suivi les étapes suivantes :

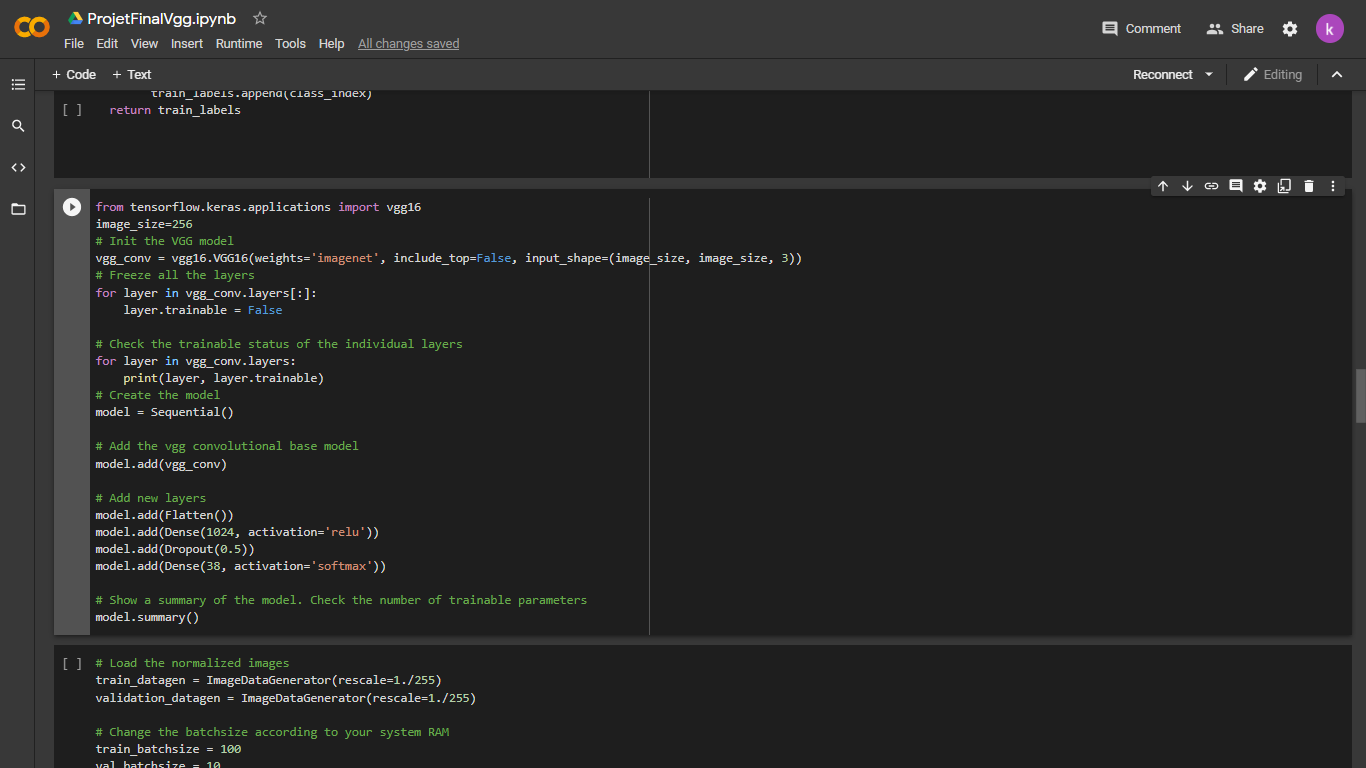
### Les importations :

J’ai importé les bibliothèques suivantes :



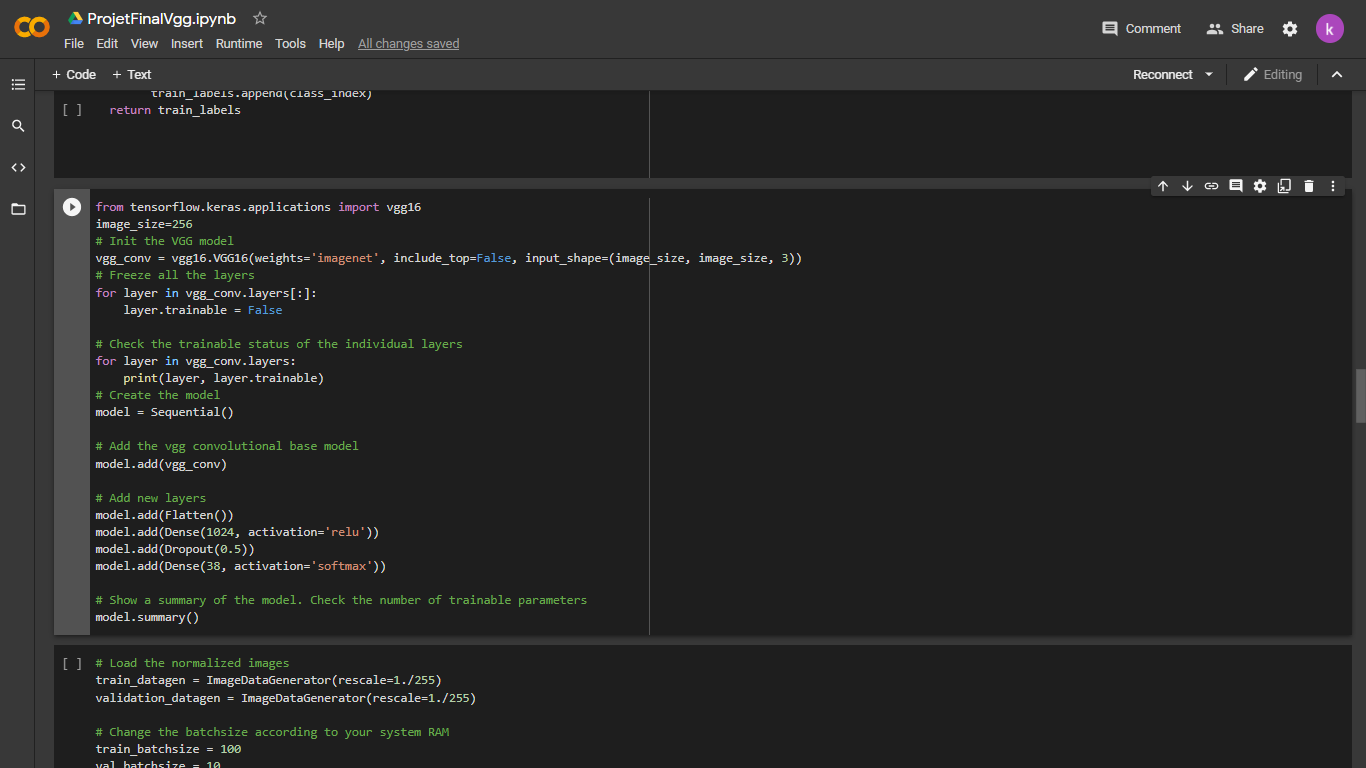
### Importation du VGG-16 de keras:

Dans cette partie de code j’ai importé le vgg-16 déjà entrainé mais sans les couches d’en haut :

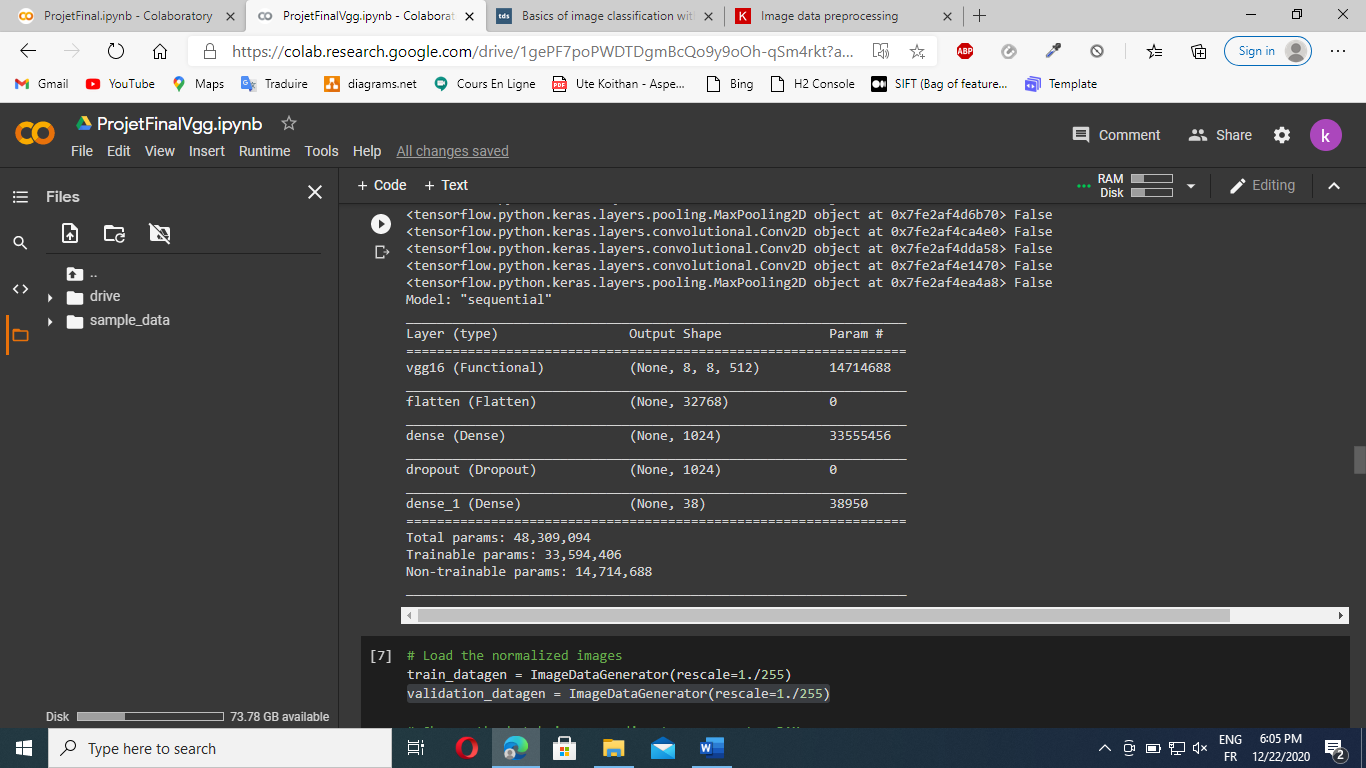


### Création du modèle :

Ici j’ai créé mon modèle, j’ai ajouté les couches de vgg-16 déjà importé , puis j’ai ajouté les couches d’en haut qui sont adaptées à ma classification :

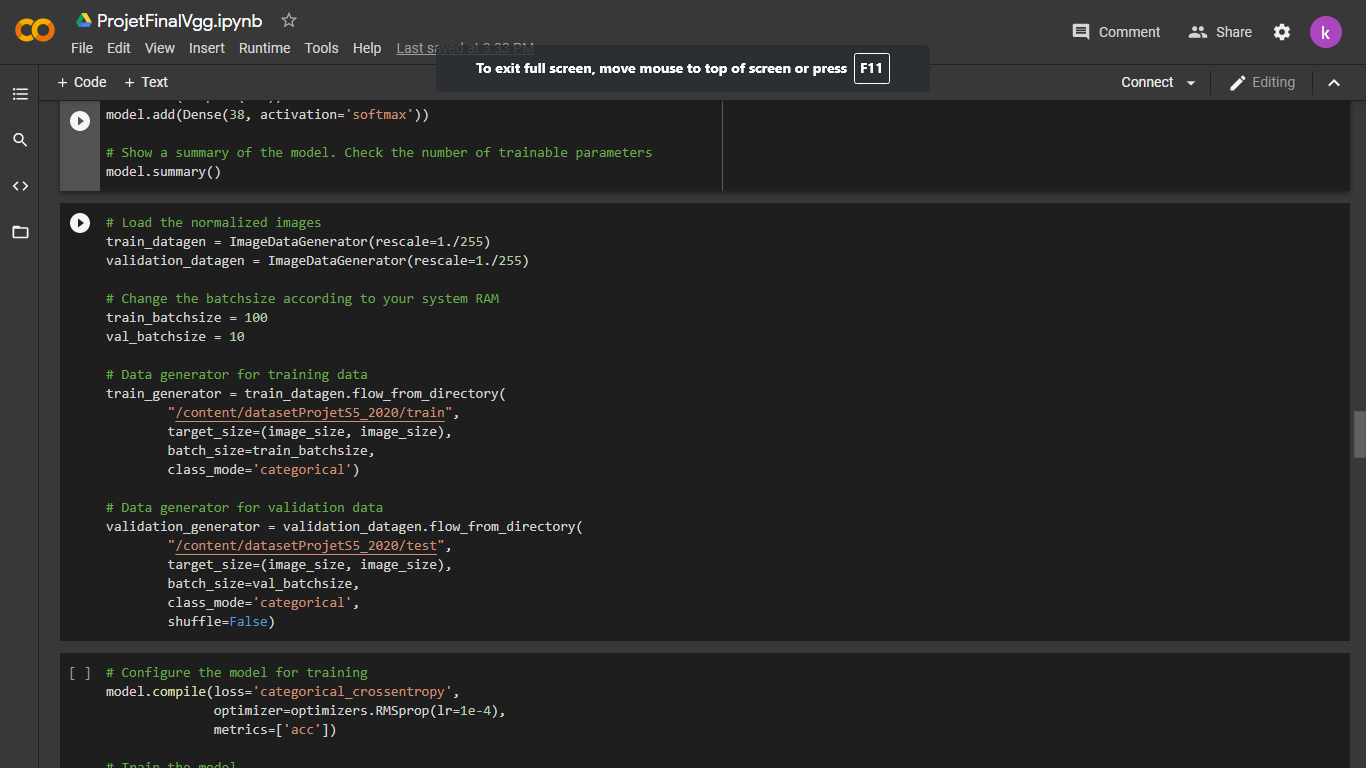


On peut remarquer qu’il ne faut pas entrainer tous les paramètres, car une partie est déjà entrainer :



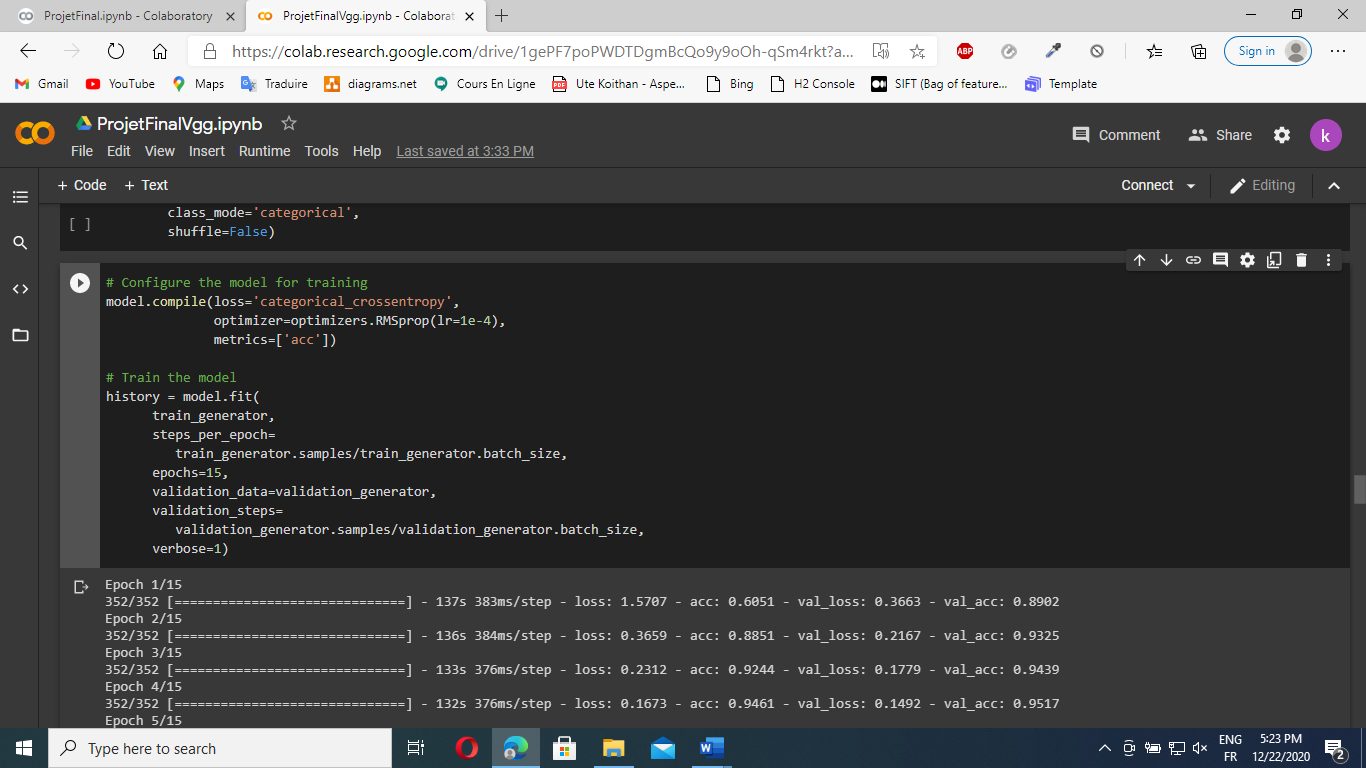
### Data/Test Generator :

On crée nos train\_generator (x\_train et y\_train) et validation\_generator (x\_test et y\_test) à partir du chemin des images :

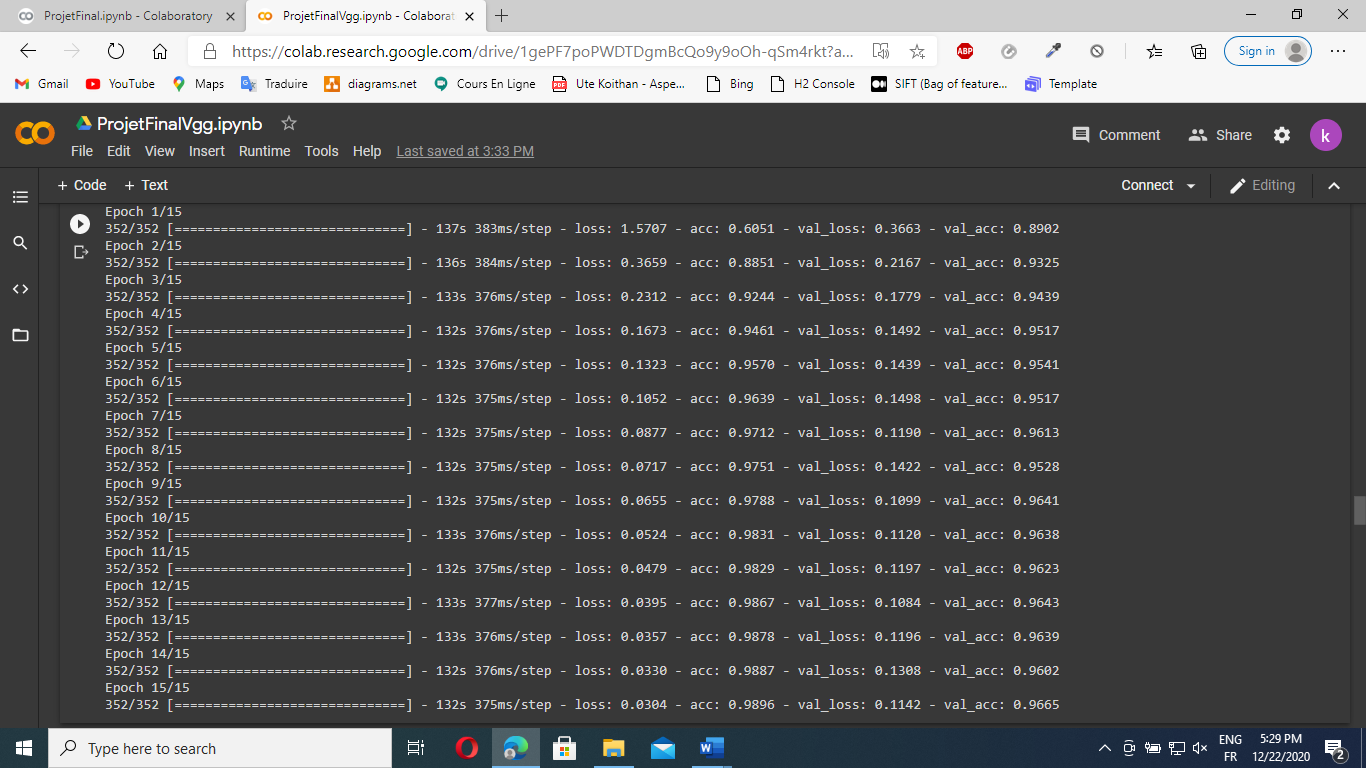


### Entrainement du modèle (couche ajouter) :

Premièrement j’ai compilé mon modèle, puis je l’ai entrainé avec 15 epoch :



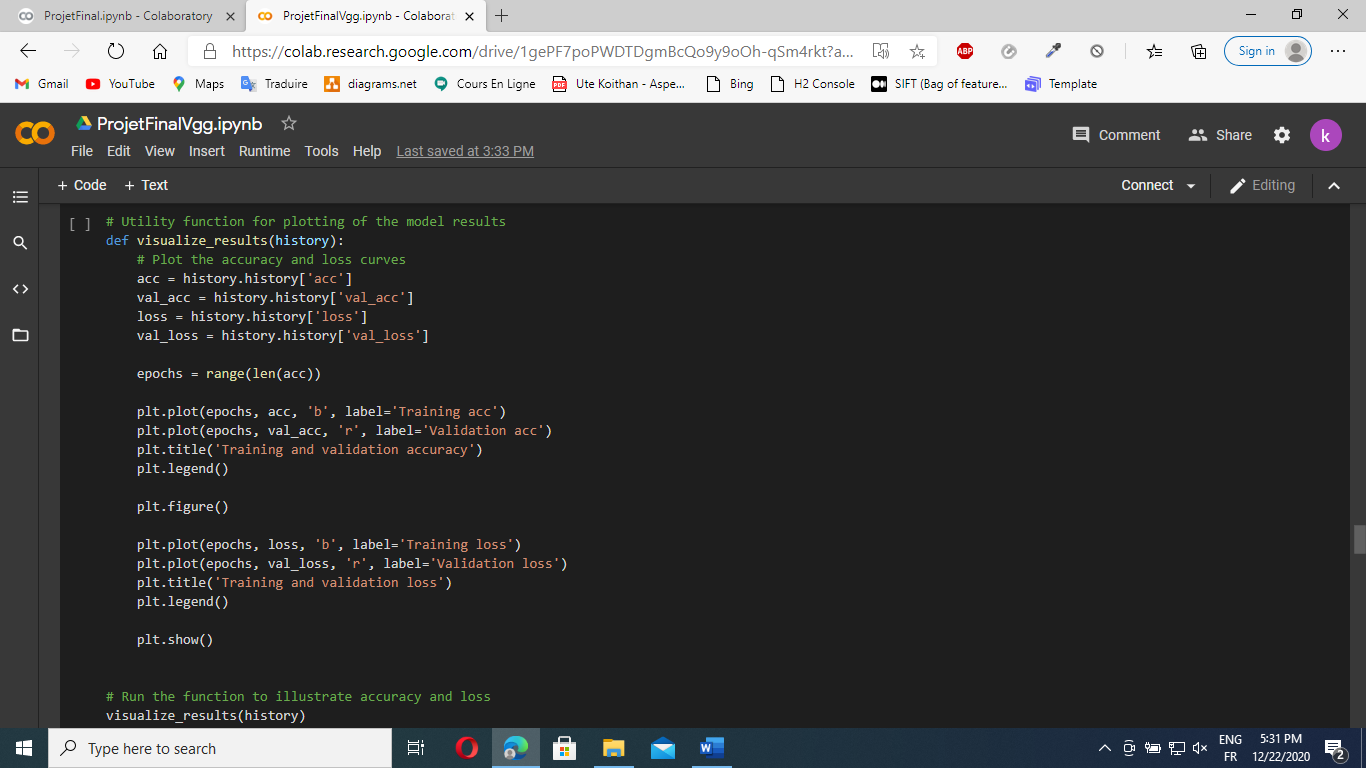
Résultats de l’entrainement :



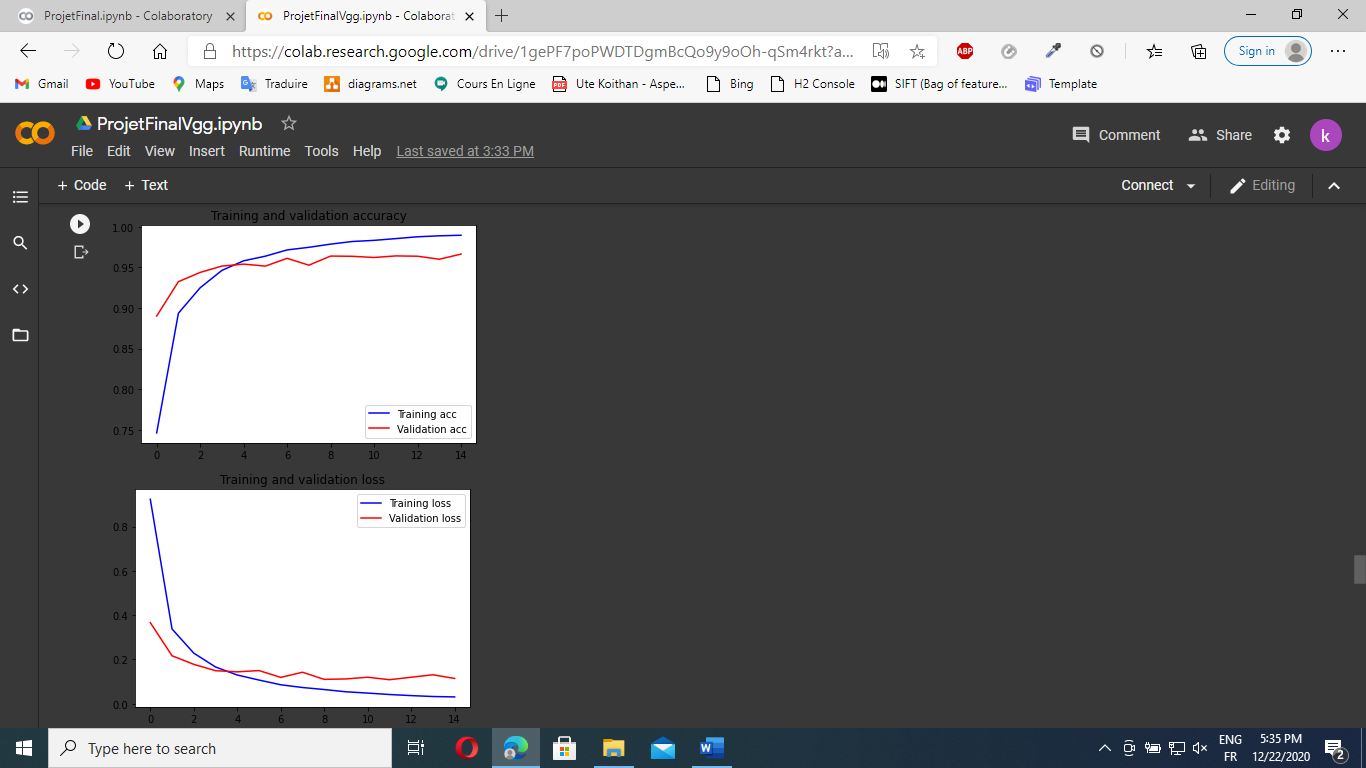
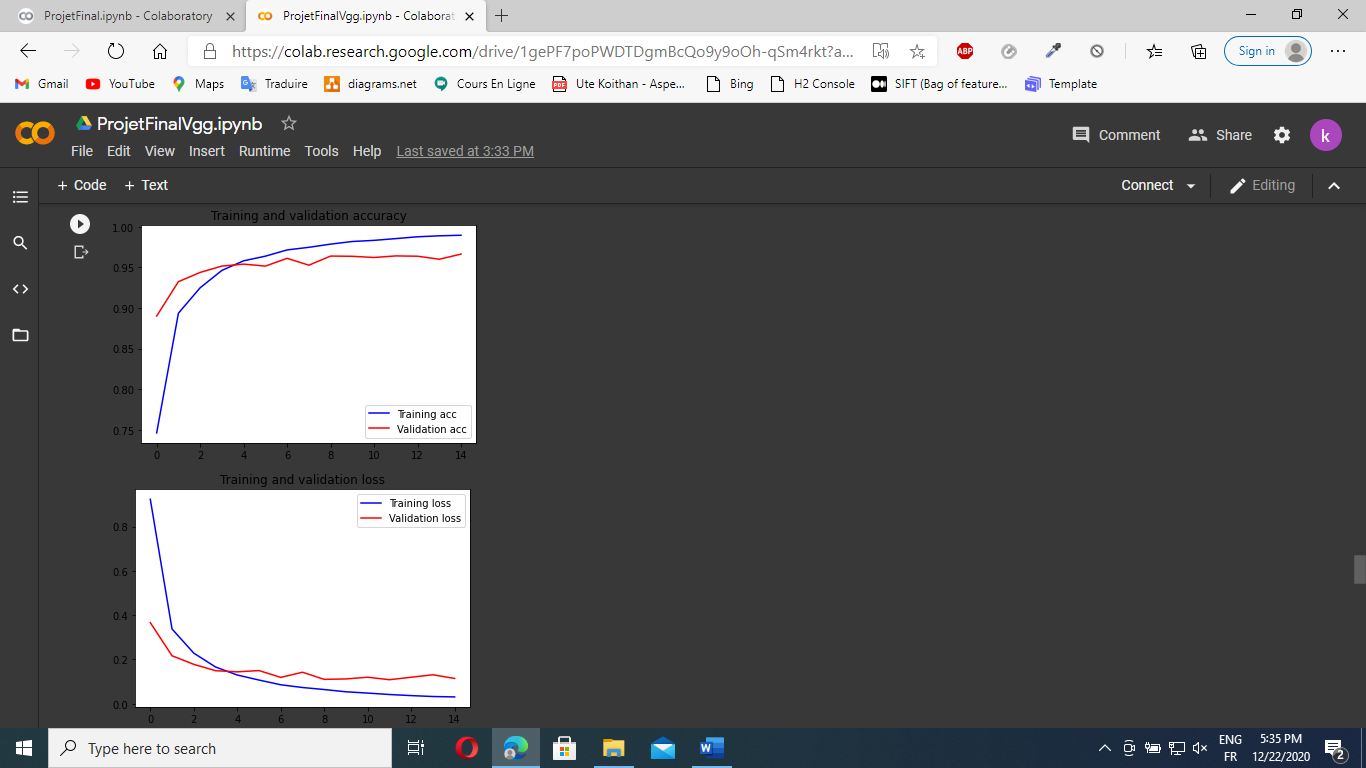
Vous pouvez remarquer le score qui augmente dans chaque epoch, il s’établie en 98% pour entrainement et 96% pour la validation

### Représentation graphique des résultats :

Voici la fonction qui sert à tracer les courbes des résultats :



Les courbes de score et d’erreur d’entrainement et de validation :

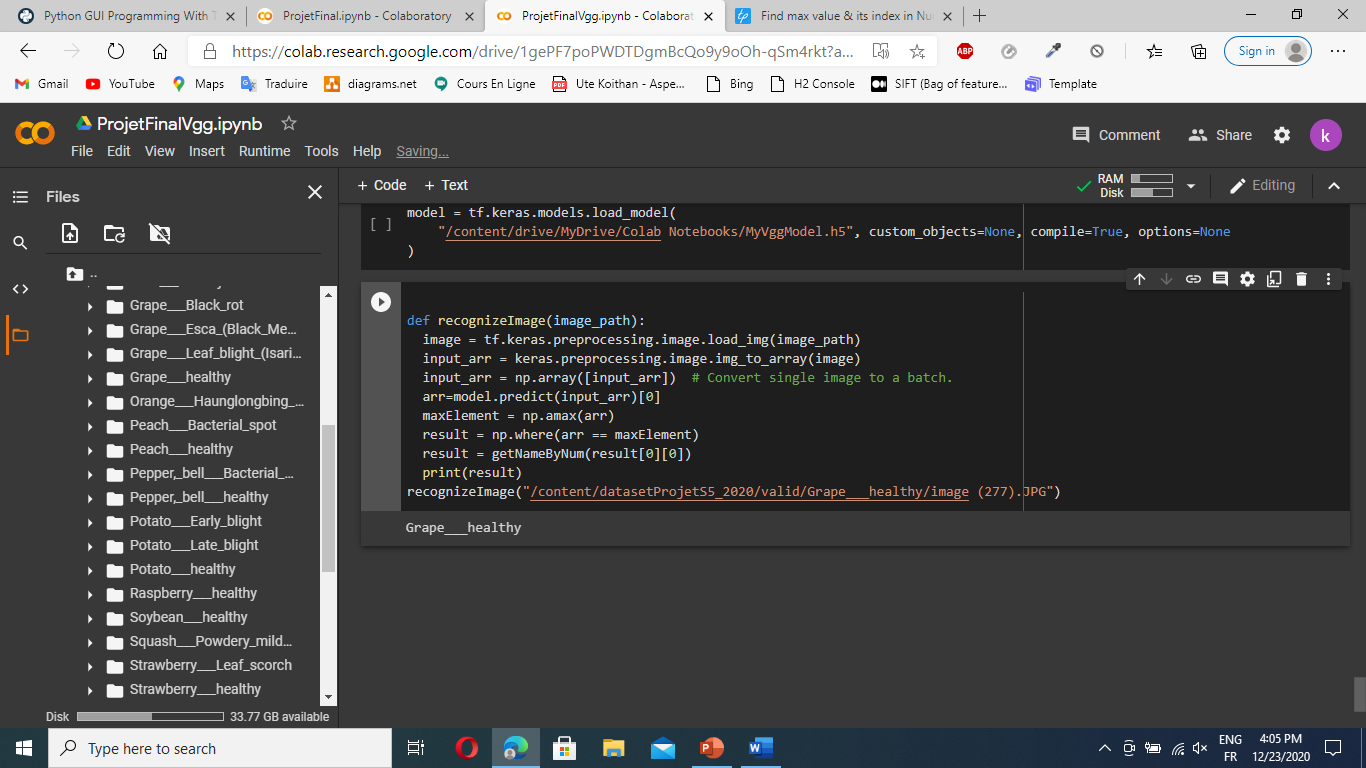


On peut remarquer que le score augmente au début brusquement et tend à se stabiliser vers la valeur 98%

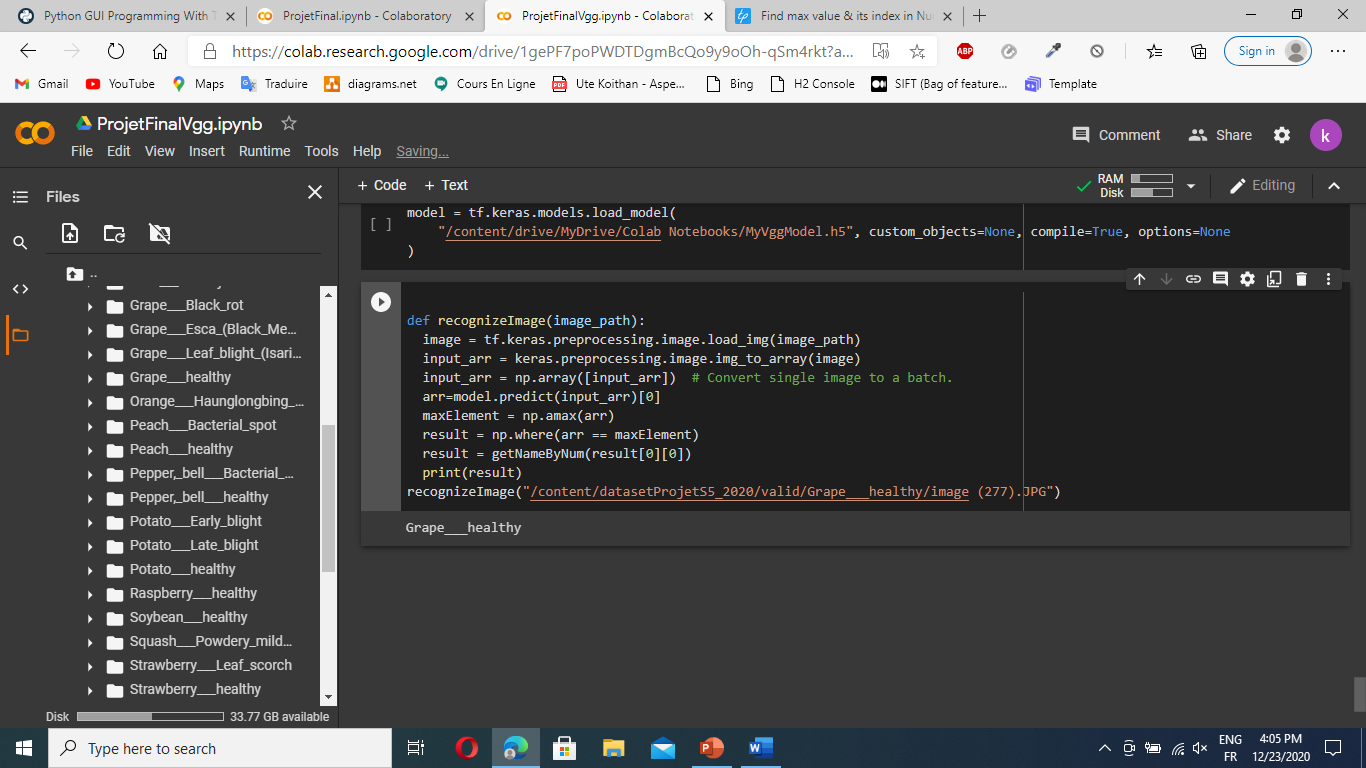
### La méthode recogniseImage(image\_path)

Cette méthode représente notre but : permettre de reconnaitre une image de feuille d’arbre.

Voici le code de la méthode :

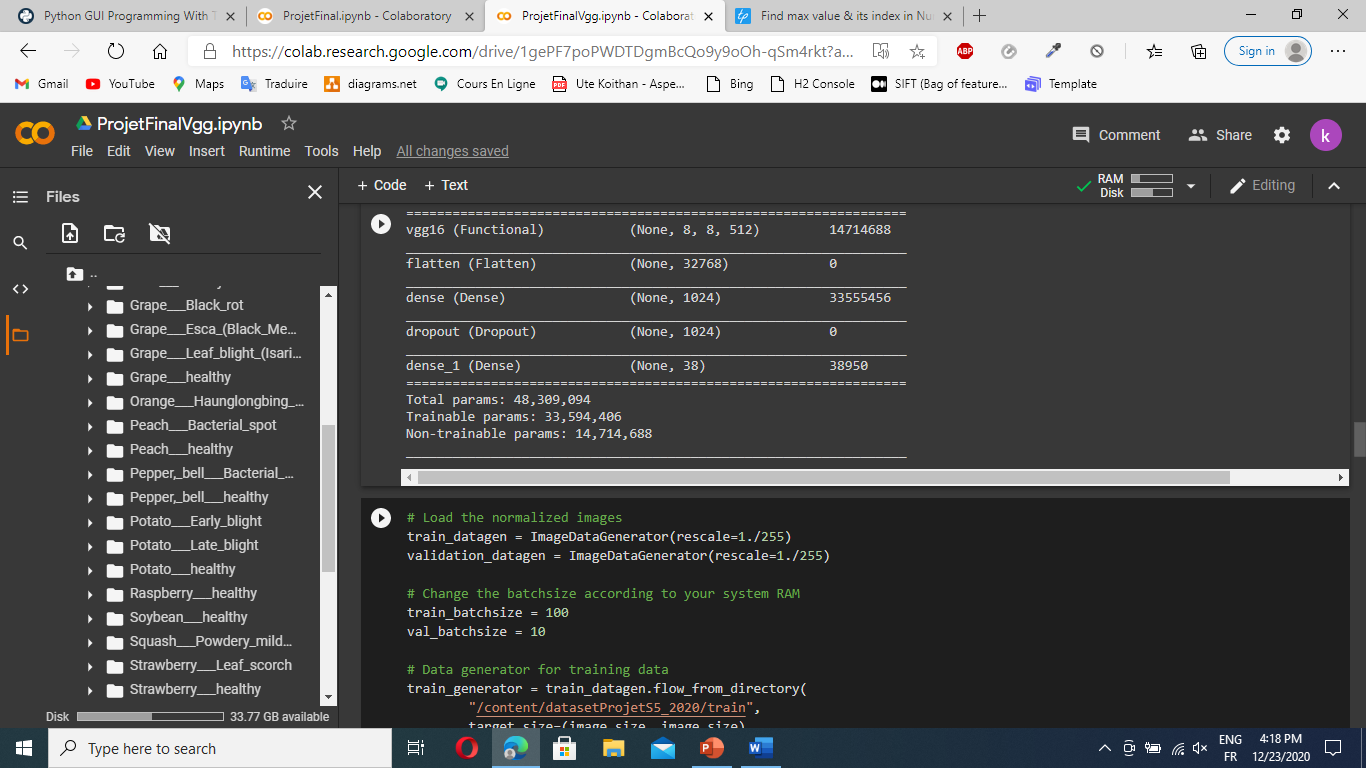


Et voici un test de la méthode :



### Discussion :

* La Première remarque est le score, on remarque que le score est très bon, 98% c’est très précis.
* Cette méthode consomme trop de ressources, pour mon cas je me suis obligé à utiliser la gpu.
* L’entrainement du modèle dans cette méthode est relativement lent.
* L’utilisation des couches vgg16 pré entrainer optimise le temps d’attente ainsi que les ressources. Vous pouvez remarquer le nombre de paramètres entrainable est moins que si j’ai à entrainer tout le modèle :

****

* On peut optimiser le score en ajoutant d’autre epoch , mais il faut faire attention , car un très grand nombre d’epoch peut conduire à un surentrainement . Comme résultat le modèle ne pourra pas reconnaitre les images qui sont un peu diffèrent des images sur lequel il est entrainé.

## **Partie 3 : Interface graphique**

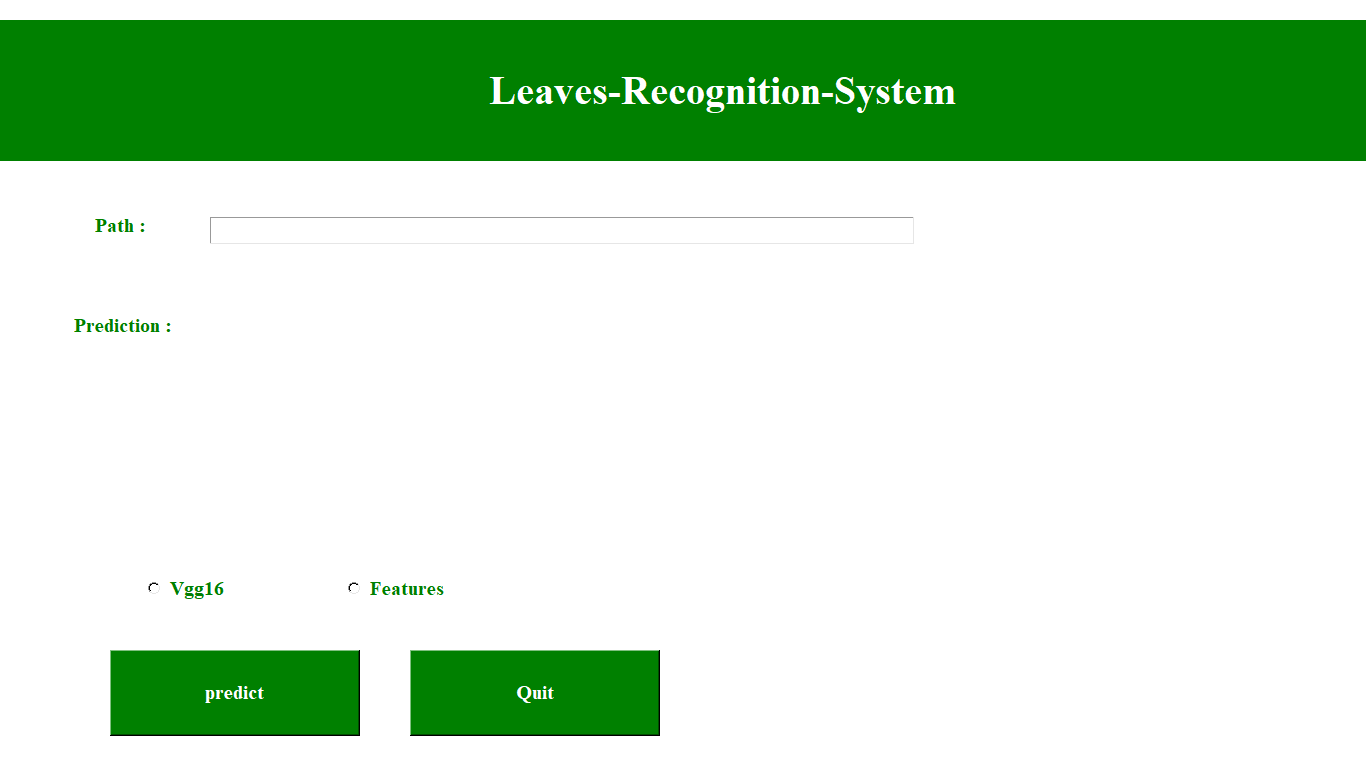
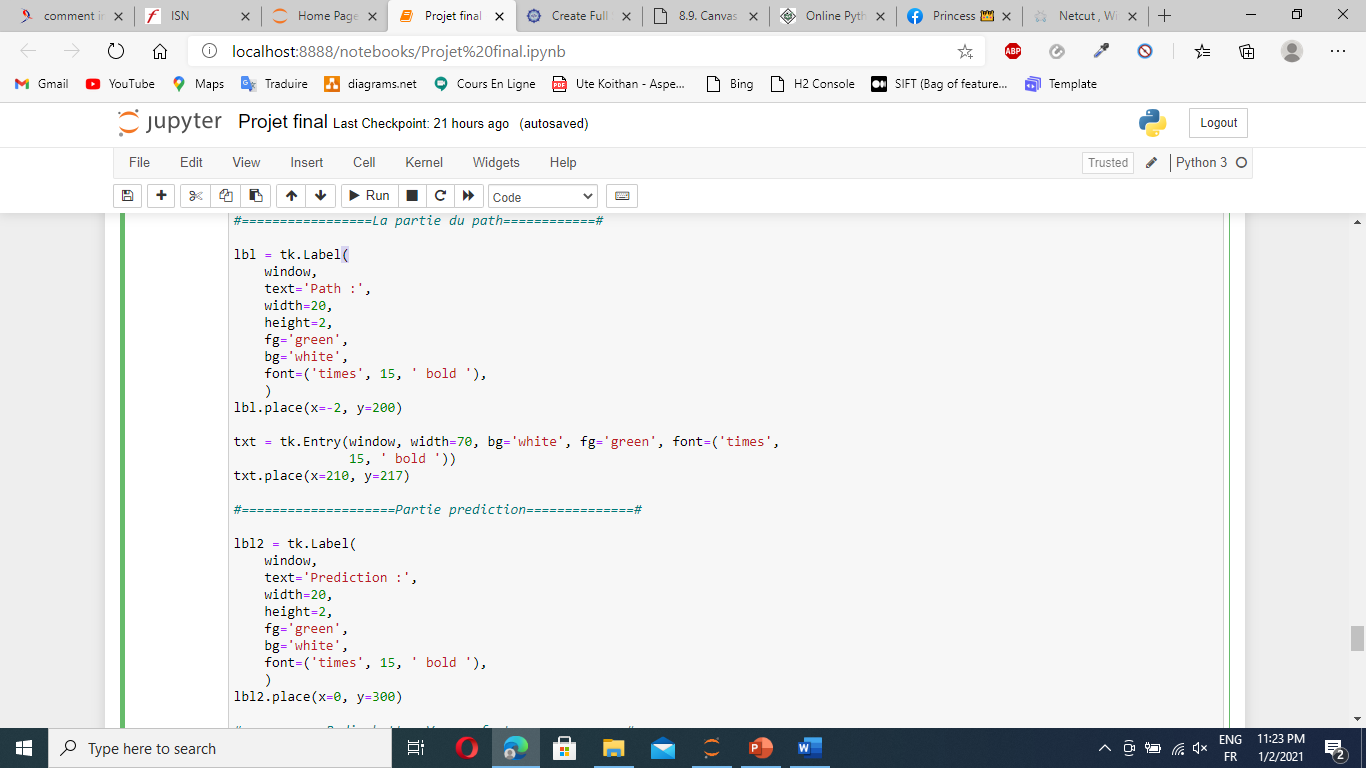
### Création de la fenêtre :

J’ai crée une fenêtre intitulé par « Leaves Recogniser » , avec un arrière-plan blanc et en mode fullscreen



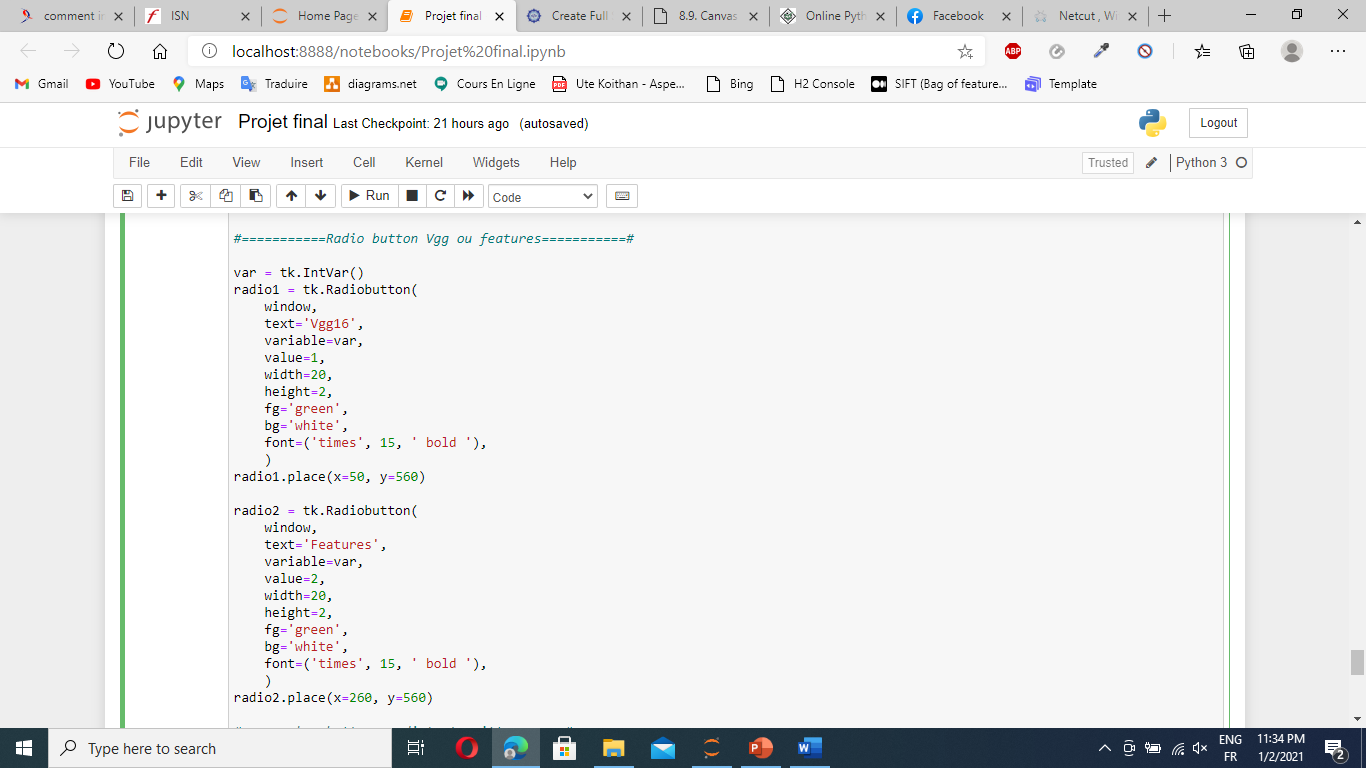
### Création des champs à remplir :

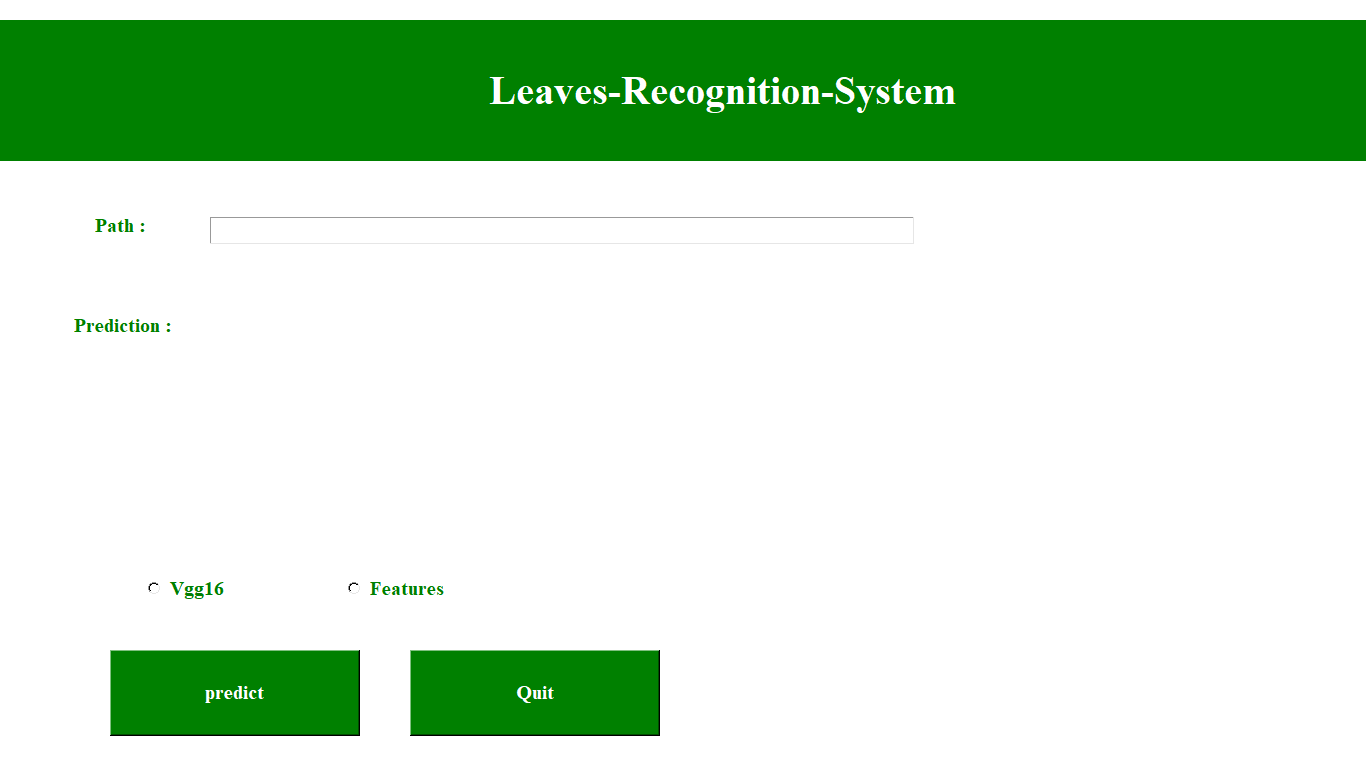
J’ai crée une label « Path » et une entry afin recevoir le chemin de la photo ou le dossier contenant des photos. Puis j’ai crée une autre label « prédiction » désignant la place réservée à contenir la prédiction des photos .



### Radio button Vgg16 / features :

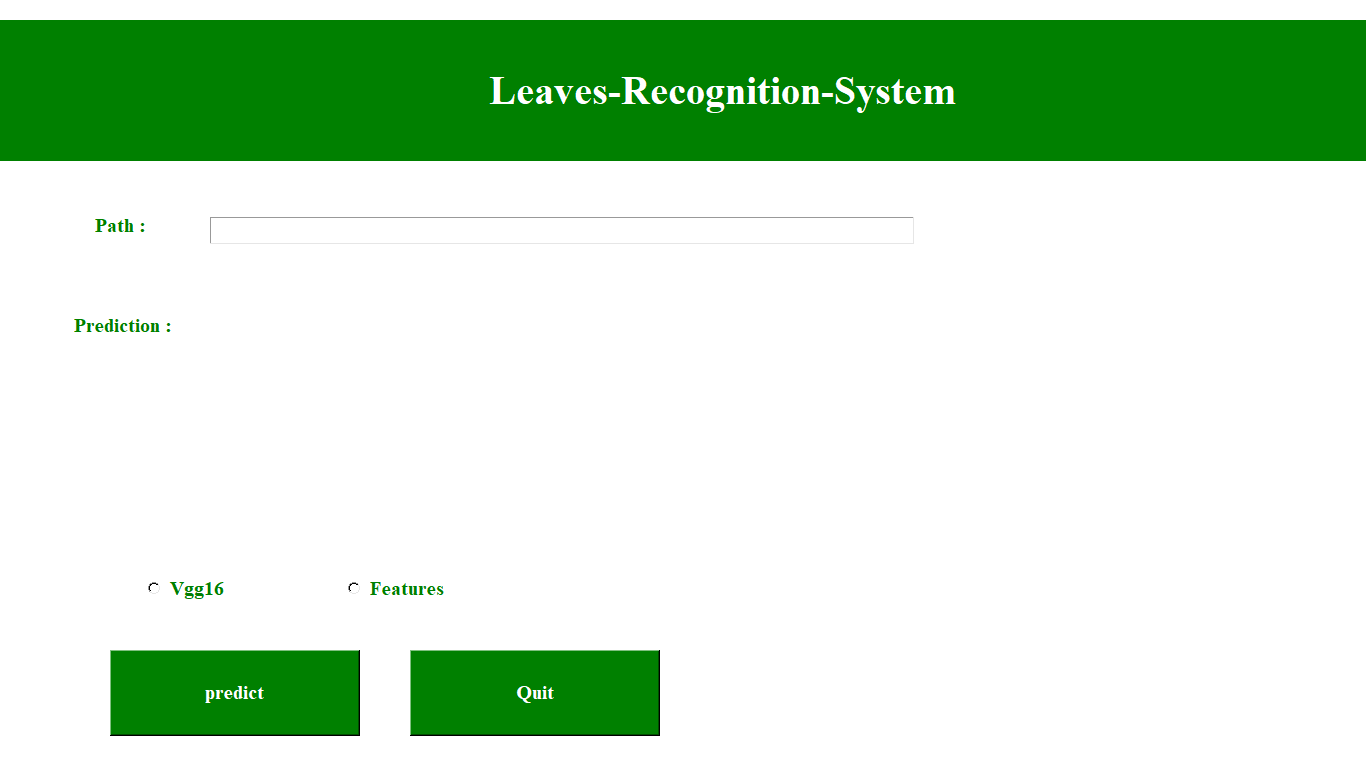
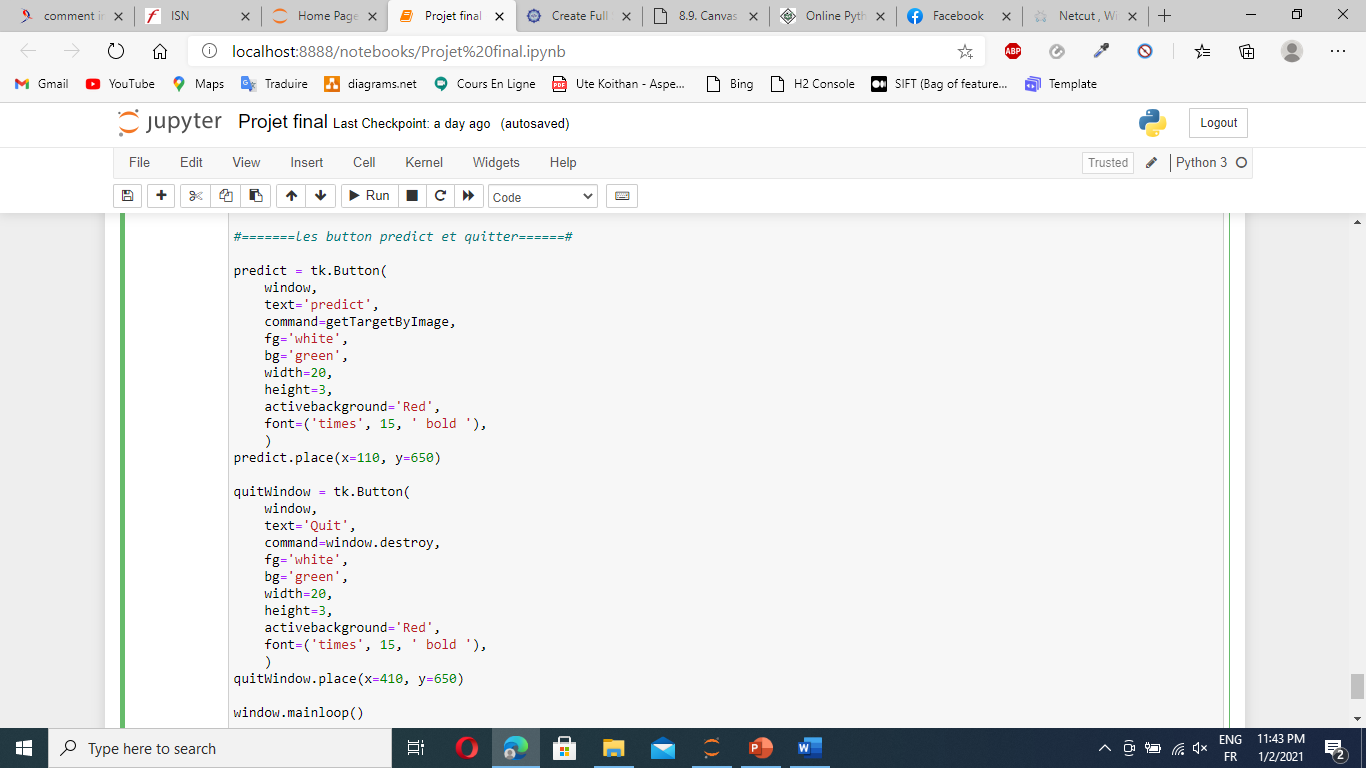
Ici j’ai crée deux Radio buttons qui permettent de choisir le modèle à utiliser (modèle de partie une ou deux)





### Création des boutons :

Ici j’ai crée une bouton « prédire » pour lancer la prédiction des images en utilisant les méthodes déjà invoquer dans les parties 1 et 2, et un bouton « quitter » qui sert à fermer la fenêtre

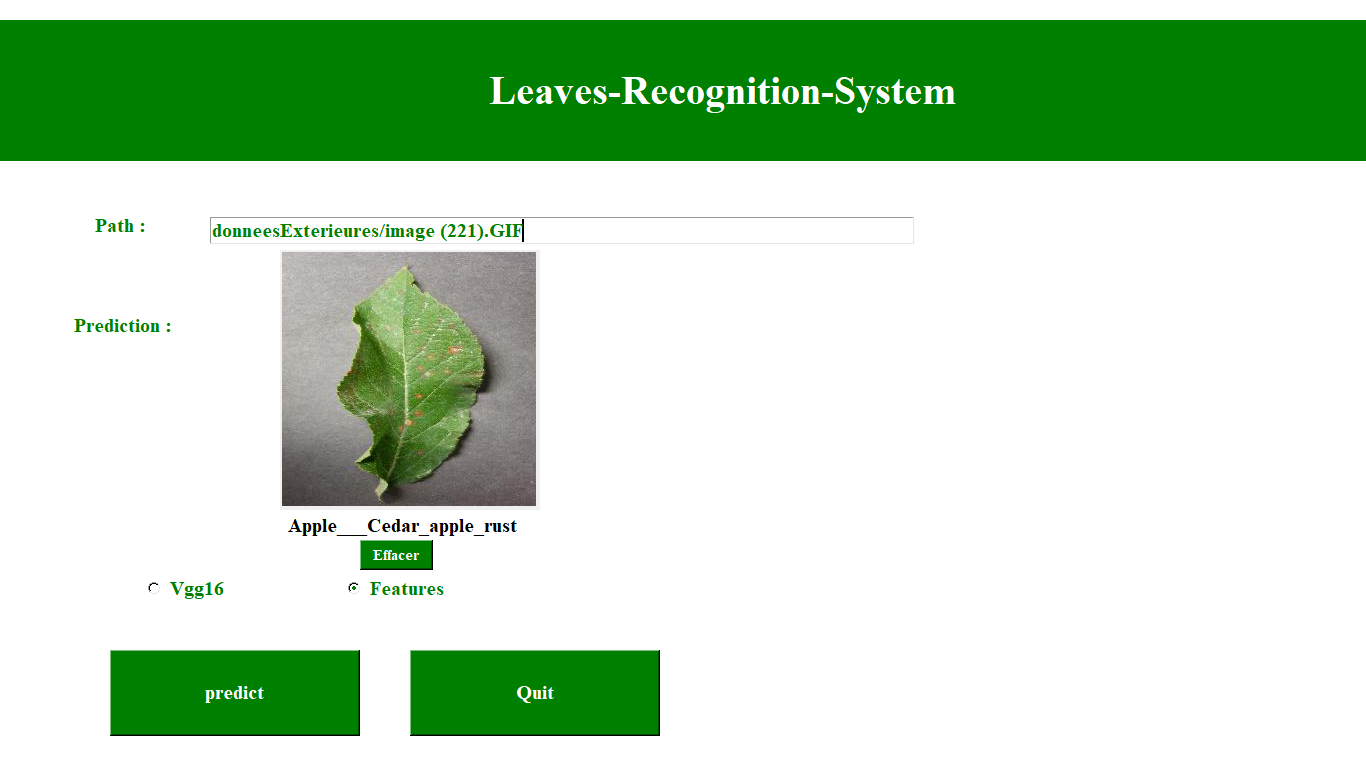


### Lancement de l’application :

Mon application peut traiter le cas d’une seule image, et le cas d’un dossier d’image :

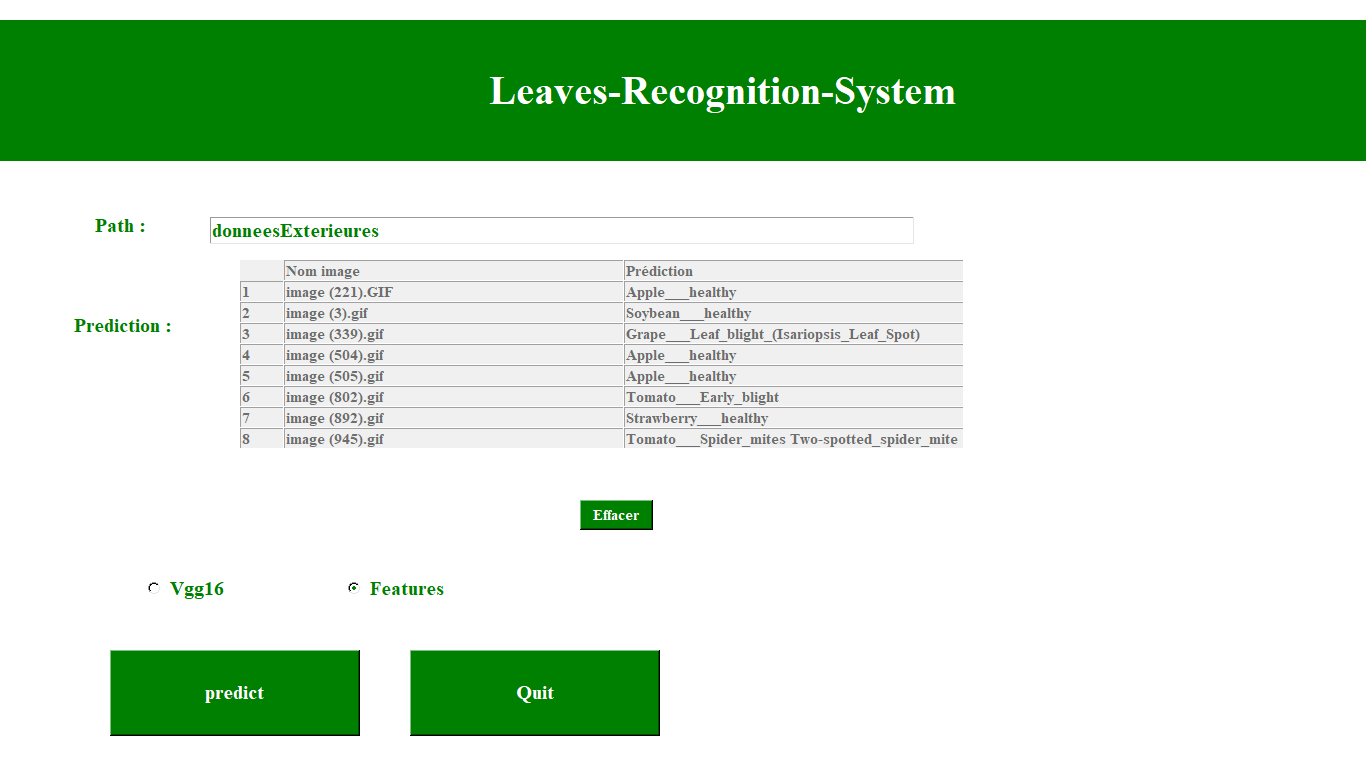
* **Le cas d’une seule image :**

J’entre le chemin d’une photo (par exemple « image (221).GIF ») puis je choisis un modèle parmi mes deux modèle (par exemple le modèle basé sur les Features), puis je clique sur predict, voici le résultat :

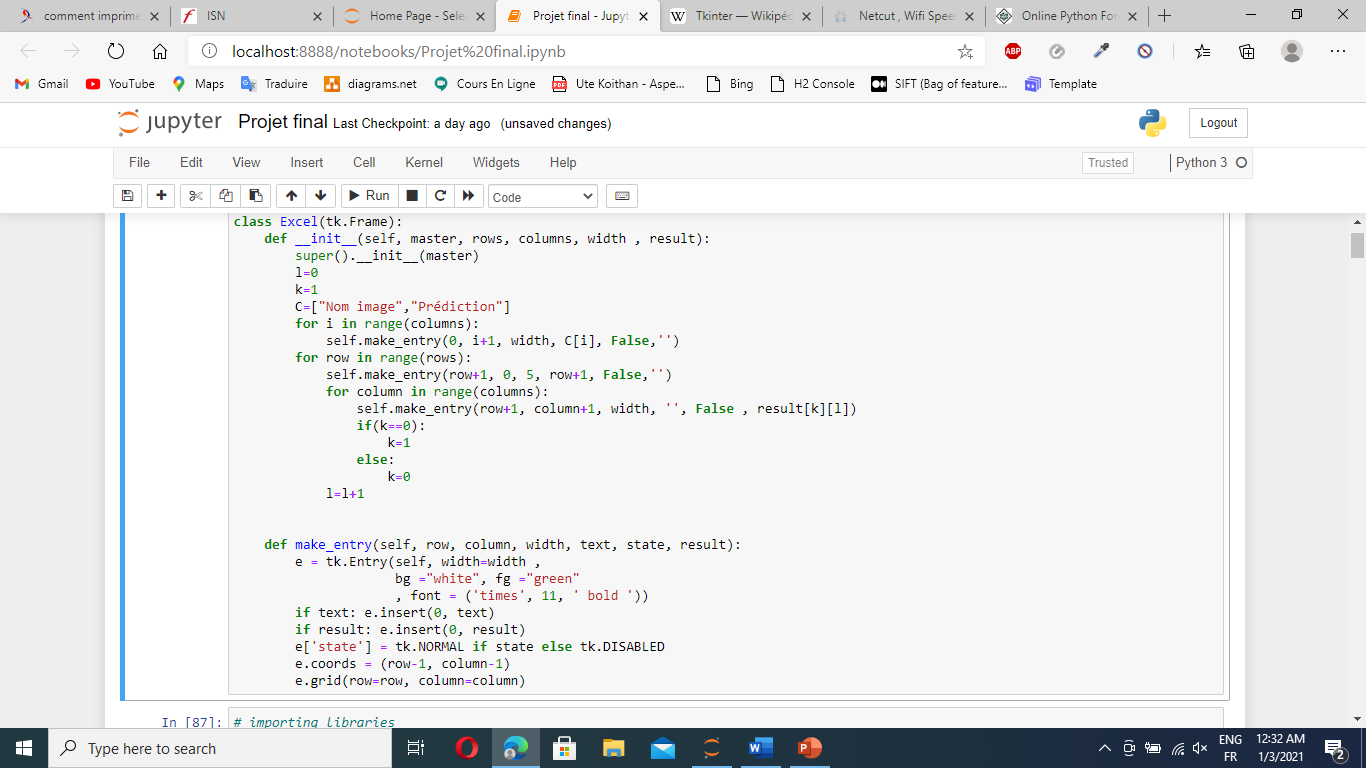


* **Le cas d’un dossier d’images :**

J’entre le chemin d’un dossier (par exemple le dossier « donneesExterieures ») puis je choisis un modèle parmi mes deux modèle (par exemple le modèle basé sur les Features), puis je clique sur predict, voici le résultat :



Pour crée le tableau ci-dessus , je me suis obligé à créer la classe « Excel » ,car il n’y a pas un widget tableau par défaut chez la bibliothèque Tkinter, alors voici la classe Excel :



Alors cette classe permet de créer un tableau dont chaque case est en vrai une « Entry » dont on peut pas écrire , le constructeur de la classe prend en arguments :

* La fenêtre dont on veut créer notre tableau
* le nombre de colonnes
* le nombre des lignes
* la taille de chaque case
* une liste qui rassemble le nom de l’image et ca classe

# Conclusion :

On peut conclure que la 2 -ème méthode utilisant les réseaux neuronaux est la plus performante, même si elle a besoin de beaucoup de ressources et de temps, mais les résultats sont vraiment très bons.

Par contre la première méthode utilisant les features, n’est pas très précis, cela peut créer des vrais soucis lors du déploiement du modèle, car une telle marge d’erreur n’est pas tolérée sur terrain.

Alors vaut mieux consommer plus de temps et de ressource sur l’entrainement (le cas de la deuxième méthode) et avoir de bon résultat, que d’entrainer rapidement un modèle qui n’est pas très fiable.

Finalement, les résultats de ce projet ont nourri ma curiosité, pour mieux connaitre le deep learning. Car j’en suis sûre que nos sociétés vont y dépendre prochainement, et grâce a lui l’humanité va accomplir d’incroyables créations et prodigieux progrès scientifiques.

# Références :

* [https://scikit-learn.org](https://scikit-learn.org/)
* [https://datascience.stackexchange.com](https://datascience.stackexchange.com/)
* [https://www.stackoverflow.com](https://www.stackoverflow.com/)
* [https://opencv.org](https://opencv.org/)
* [https://github.com](https://github.com/)
* [https://openclassrooms.com](https://openclassrooms.com/)
* <https://Wikipedia.org>