**Instructions Docker pour le Gradio**

Création du répertoire de travail

Dans la machine virutelle gcloud, j’ai seulement les dossiers suivants (pour le moment) :

|--- Workspace

| |--- Defi-IA-bedbugs

| | |--- Dockerfile

| | |--- gradio\_defiIA.py

| | |--- submissions.csv

| | |--- dictionnary

| | | |--- child

| | | |--- mobile

| | | |--- parking

| | | |--- pool

| | | |--- target\_encoding

| | |--- models\_sav

| | | |--- 2000\_XGboost\_target\_encoding\_12-11.sav

Cette machine virtuelle représente la machine locale du prof qui doit être sous Linux.

Rq : Je conseillerais de ne pas avoir trop de fichiers dans gcloud, car j’ai vite eu plus de place. Je conseille de ne garder que les fichiers nécessaires au gradio ci-dessus et de supprimer les précédentes images ou containers. J’ai marqué les commandes en jaune dans le dernier encadré plus bas.

Dockerfile :

# We use a minimal Ubuntu image optimized for Docker

FROM ubuntu:latest

# Set up for your local zone an UTC information

ENV TZ=Europe/Paris

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

# Our own build instructions

RUN apt-get update && apt-get install -y \

#git \

#wget \

#nano \

python3-pip

# Additional librairies:

RUN pip install gradio==3.4

RUN pip install xgboost==1.0.1

RUN pip install numpy==1.23.3

RUN pip install pandas==1.5.0

RUN pip install scikit-learn==1.1.2

RUN pip install seaborn==0.12.1

RUN pip install sklearn==0.0

RUN pip install category-encoders==2.5.1.post0

RUN pip install matplotlib==3.6.0

RUN pip install scipy==1.9.1

RUN pip install pickle5

# Clean up APT when done.

RUN apt-get clean && rm -rf /var/lib/apt/lists/\* /tmp/\* /var/tmp/\*

* Se référer au fichier Docker\_avec\_gcloud.docx pour comprendre comment obtenir ceci sur gcloud.

Commandes à réaliser pour lancer le Gradio.

* Seules les commandes en gris sont vraiment essentielles à exécuter.
* Dans gcloud, se placer dans le bon répertoire

cd Workspace/Defi-IA-bedbugs

**1**

* Créer une image du Dockerfile

sudo docker build -t image1 .

- Le point « . » indique l’emplacement du Dockerfile. Ici, le Dockerfile se trouve dans le dossier courant.

- Commande générale : sudo docker build -t name\_image path\_dockerfile

**2**

*Rq: Lors de la création de l’image le warning suivant apparaît:*

*WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead:* [*https://pip.pypa.io/warnings/venv*](https://pip.pypa.io/warnings/venv)

* *Ceci vient du fait que la façon dont notre conteneur est construit n'ajoute pas d'utilisateur, donc tout est fait en tant que root. On peut créer un utilisateur et installer dans le répertoire personnel de cet utilisateur. Voir ce lien pour le faire :* [*https://stackoverflow.com/questions/68673221/warning-running-pip-as-the-root-user*](https://stackoverflow.com/questions/68673221/warning-running-pip-as-the-root-user)
* Vérifier que l’image existe en listant les images crées :

sudo docker image ls

**3**

* Créer un container et mount le volume de **Defi-IA-bedbugs** dans le conteneur :

docker run -it --name container1 -v "$(pwd)":/mnt image1

- La commande run est équivalente aux commandes create (pour créer un conteneur) et start (pour activer un conteneur).

- Le flag -it permet de rentrer dans le mode interactif du conteneur.

- Avec --name container1, le container se nomme container1

- Le flag -v permet de mount un volume dans le container i.e. on met le contenu du dossier **Defi-IA-bedbugs** dans le dossier **mnt** du container.

- Comme nous nous situons déjà dans le dossier **Defi-IA-bedbugs**, le chemin absolu de ce dossier est donné par "$(pwd)". Le chemin absolu du dossier **mnt** appartenant au conteneur est donné par /mnt

- Commande générale :

docker run -it --name name\_container -v \

absolute\_path\_host\_foler:absolute\_path\_container\_folder

- Commande pour créer un conteneur sans mount :

sudo docker run -it --name container1 image1

**4**

* Quelques commandes sur les conteneurs :

docker start container1 #active un conteneur

docker attach container1 #entre dans le mode interactif du conteneur activé

exit #sort du conteneur

doker stop container1 #désactive le conteneur

docker container rm container1 container2 #supprime container1 et container2

docker container rm -vf $(docker container ls -aq) #supprime tous les containers

docker image rm image2 image2 #supprime image1 et image2

docker rmi -f $(docker images -aq) #supprime toutes les images

* A l’issue de la commande ci-dessus, on est normalement dans le conteneur.

On se place dans le dossier **mnt** du conteneur :

cd mnt

ls #pour vérifier que l’on a bien le contenu de **Defi-IA-bedbugs** dans **mnt**

**5**

* Lancer le gradio :

**python3 gradio\_defiIA.py --model\_name "2000\_XGboost\_target\_encoding\_12-11.sav"**

**6**

* Vérifier l’état des conteneurs

sudo docker container ls #liste les conteneurs actifs

sudo docker container ls -a #liste tous les conteneurs

**7**

Prochaines étapes :

Je pense que l’on peut procéder de la sorte :

Le prof devra :

* Cloner notre **git** dans sur sa machine locale. Le dossier cloné s’appellera **Defi-IA-bedbugs** (je crois).
* Toujours sur sa machine locale, il devra chercher les poids du modèle avec **wget** et les placer dans le dossier **Defi-IA-bedbugs**. On se retrouve dans une configuration à peu près similaire à ce que j’avais sur mon gcloud (sauf que le prof n’a pas le dossier Workspace).
* Télécharger le Dockerfile (si pas déjà fait) et le placer dans le dossier **Defi-IA-bedbugs**
* Ouvrir un terminal de commande Docker et se placer dans le dossier **Defi-IA-bedbugs** avec un cd.

*Rq : Je n’ai pas encore installé Docker sur ma machine (par manque de place, je fais tout sur gcloud), donc j’imagine qu’il a un terminal de commande docker, sinon ouvrir un terminal normal j’imagine.*

* On se retrouve à l’encadré **2** (Créer une image du Dockerfile)
* Faire les étapes 2 à 3.
* **/.\** Lors de l’étape 3, il ne faudra peut-être pas tout mount dans le container, mais uniquement ce qui est utile car ça risque de prendre trop de place.

Il faudra peut-être créer un dossier « to\_mount » dans **Defi-IA-bedbugs (ou directement dans le git)** qui contiendra uniquement les fichiers à mount.

* Finir les étapes 3-7
* Rajouter des étapes pour run les modèles (quand on aura transformé les ipynb en py)