Atelier 1 : Installation de l'environnement

Prérequis : Notions de base en machine learning et un script Jupyter fonctionnel (votre projet ML du premier semestre).

1. **Objectifs :**
   I. Installer un environnement de développement sous WSL (Windows Subsystem for Linux) ou sur vmware.
   II. Installer le système d'exploitation ubuntu
   III. Configurer Python 3 et les outils nécessaires.
2. **Étapes :**
   I. Installation de WSL et Ubuntu.
   II. Installation de Python 3, pip, et virtualenv.
   III. Création d'un environnement virtuel.
   IV. Installation des dépendances (via requirements.txt préparé qui contient tous les dépendances à installer de votre projet).
3. **Livrables :** Un environnement Python sur une machine ubuntu prêt pour l'exécution de scripts.

Tutoriel : Installation de l'environnement pour un projet de Machine Learning sous WSL

I. Étape 01: Activer WSL (Windows Subsystem for Linux) sur windows:

1) Ouvrir PowerShell en mode Administrateur :

Cliquer droit sur le bouton Démarrer → CMD Windows (Admin) ou PowerShell (Admin).

Exécuter la commande wsl --update

2) Activer WSL : Exécuter la commande « wsl --install»

```
PS C:\WINDOWS\system32> wsl --install
Installation : Ubuntu
[                          0,0%                          ]
```

Cela installe automatiquement WSL 2 et la dernière version d'Ubuntu.

Pour installer une version précise d'ubuntu, vous pouvez tapez la commande suivante : wsl --install -d Ubuntu-20.04. Pour vérifier l'installation, lancer la commande

« wsl --version »

```
PS C:\WINDOWS\system32> wsl --version
Version WSL : 2.0.14.0
Version du noyau : 5.15.133.1-1
Version WSLg : 1.0.59
Version MSRDC : 1.2.4677
Version direct3D : 1.611.1-81528511
Version de DXCore : 10.0.25131.1002-220531-1700.rs-onecore-base2-hyp
Version de Windows : 10.0.22631.4460
```

II. Étape 02: Installer Ubuntu

1) Une fois l'installation terminée, redémarrer la machine si demandée, puis lancer Ubuntu avec la commande « wsl »

```
PS C:\WINDOWS\system32> wsl
```

Cela ouvrira la console Ubuntu et vous demandera de créer un utilisateur et un mot de passe pour finaliser la configuration.

En cas de plusieurs OS sur wsl, tapez la commande suivante en précisant la version à ouvrir (Ubuntu-20.04 dans ce cas) :

wsl -d Ubuntu-20.04

```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: sirine
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Dec 23 22:19:37 CET 2024

  System load:  0.73               Processes:             67
  Usage of /:   0.1% of 1006.85GB  Users logged in:       0
  Memory usage: 4%                 IPv4 address for eth0: 172.20.175.178
  Swap usage:   0%


This message is shown once a day. To disable it please create the
/home/sirine/.hushlogin file.
sirine@DESKTOP-4U94PKC:~$
```

III. Étape 03: Installer Python 3, pip et virtualenv

1) Mettre à jour les paquets :

« sudo apt update && sudo apt upgrade -y »

```
sirine@DESKTOP-4U94PKC:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for sirine:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2006 kB]
0% [Waiting for headers] [3 Packages 12.3 kB/2006 kB 1%]
```

2) Installer Python 3 et pip :

« sudo apt install python3 python3-pip -y »

```
sirine@DESKTOP-4U94PKC:~$ sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.10.6-1~22.04.1).
python3 set to manually installed.
The following additional packages will be installed:
```

3) Vérifier les versions installées :

« python3 --version »

« pip3 --version »

```
sirine@DESKTOP-4U94PKC:~$ python3 --version
Python 3.10.12
sirine@DESKTOP-4U94PKC:~$ pip3 --version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)
```

4) Installer virtualenv pour créer des environnements virtuels :

« sudo apt install python3-virtualenv »

```
sirine@DESKTOP-4U94PKC:~/ml_project$ sudo apt install python3-virtualenv
[sudo] password for sirine:
Sorry, try again.
[sudo] password for sirine:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-distlib python3-filelock python3-pip-whl python3-platformdirs python3-setuptools-whl python3-wheel-whl
```

5) Créer un environnement virtuel

A- Créer un dossier pour votre projet :

« mkdir ~/ml_project && cd ~/ml_project »

```
sirine@DESKTOP-4U94PKC:~$ mkdir ~/ml_project && cd ~/ml_project
sirine@DESKTOP-4U94PKC:~/ml_project$
```

NB : Le nom du dossier doit **obligatoirement** respecter cette forme :

prénom-nom-classe-ml_project

B- Créer un environnement virtuel :

« virtualenv venv »

```
sirine@DESKTOP-4U94PKC:~/ml_project$ virtualenv venv
created virtual environment CPython3.10.12.final.0-64 in 402ms
  creator CPython3Posix(dest=/home/sirine/ml_project/venv, clear=False, no_vcs_ignore=False,
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, a
    added seed packages: pip==24.3.1, setuptools==75.6.0, wheel==0.45.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator
```

C- Activer l'environnement virtuel :

« source venv/bin/activate »

```
sirine@DESKTOP-4U94PKC:~/ml_project$ source venv/bin/activate
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$
```

Lorsque l'environnement est actif, le terminal affichera (venv) au début de chaque commande.

✓ Ajouter les permissions d'exécution (si nécessaire)

Si vous rencontrez des problèmes de permissions, donnez les droits nécessaires au script d'activation :

« chmod +x venv/bin/activate »

```
sirine@DESKTOP-4U94PKC:~/ml_project$ chmod +x venv/bin/activate
```

IV. Étape 04: Installer les dépendances

1) Préparer le fichier « requirements.txt » :

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ nano requirements.txt
```

Voici un exemple de contenu pour ce fichier (modifiable selon les bibliothèques utilisées dans votre projet):

« numpy, pandas, scikit-learn, matplotlib, jupyter, mlflow, fastapi, uvicorn, joblib »

```
  GNU nano 6.2                                    requirements.txt *
numpy
pandas
scikit-learn
matplotlib
jupyter
mlflow
fastapi
uvicorn
joblib
```

Le fichier « requirements.txt » est utilisé dans un projet Python pour lister les bibliothèques ou dépendances nécessaires afin de garantir que l'environnement du projet est configuré correctement.

➔ Voici une explication brève pour chaque bibliothèque listée dans **requirements.txt** :

- **numpy** : Permet des calculs rapides sur des tableaux multidimensionnels.
- **pandas** : Fournit des outils pour manipuler et analyser des données structurées.
- **scikit-learn** : Implémente des algorithmes de machine learning et des outils d'évaluation.

- **matplotlib** : Utilisé pour créer des graphiques et visualiser des données.
- **jupyter** : Plateforme interactive pour exécuter et documenter du code Python.
- **mlflow** : Aide à suivre, gérer et déployer les modèles de machine learning.
- **fastapi** : Framework rapide pour construire des API web modernes.
- **uvicorn** : Serveur performant pour exécuter des applications FastAPI.
- **joblib** : Sauvegarde et charge efficacement des modèles ou données volumineux.

2) Installer les dépendances :

« pip install -r requirements.txt »

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ pip install -r requirements.txt
Collecting numpy (from -r requirements.txt (line 1))
  Downloading numpy-2.2.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (62 kB)
Collecting pandas (from -r requirements.txt (line 2))
  Downloading pandas-2.2.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (89 kB)
Collecting scikit-learn (from -r requirements.txt (line 3))
  Downloading scikit_learn-1.6.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Collecting matplotlib (from -r requirements.txt (line 4))
  Downloading matplotlib-3.10.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting jupyter (from -r requirements.txt (line 5))
  Downloading jupyter-1.1.1-py2.py3-none-any.whl.metadata (2.0 kB)
Collecting mlflow (from -r requirements.txt (line 6))
  Downloading mlflow-2.19.0-py3-none-any.whl.metadata (30 kB)
```

3) Vérifier l'installation des paquets :

« pip list »

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ pip list
Package                           Version
--------------------------------- -------------
alembic                           1.14.0
annotated-types                   0.7.0
anyio                             4.7.0
argon2-cffi                       23.1.0
argon2-cffi-bindings              21.2.0
arrow                             1.3.0
asttokens                         3.0.0
async-lru                         2.0.4
attrs                             24.3.0
babel                             2.16.0
beautifulsoup4                    4.12.3
bleach                            6.2.0
blinker                           1.9.0
cachetools                        5.5.0
```

V. Tester l'environnement

1) Créer un fichier de test (exemple : test_environment.py) :

```
sirine@DESKTOP-4U94PKC:~/ml_project$ nano test_environment.py
```

```
  GNU nano 6.2
import numpy as np
import pandas as pd
import sklearn
import mlflow

print("Environment is set up correctly!")
```

```
import numpy as np
import pandas as pd
import sklearn
import mlflow
print("Environment is set up correctly!")
```

2) Exécuter le script :

« python test_environment.py »

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ python test_environment.py
Environment is set up correctly!
```

Livrables

1) Dossier de projet contenant :

a. Un environnement virtuel (venv).

b. Un fichier requirements.txt.

c. Un script de test (test_environment.py).

2) Un terminal prêt pour exécuter des scripts Jupyter ou autres :

Pour lancer Jupyter Notebook dans cet environnement :

« jupyter notebook »

⇨ Si la bibliothèque n'est pas installée, vous devez lancer la commande « pip install notebook »

3) Vérification de l'environnement :

Vous devez lancer la commande « pip freeze » e pour lister les paquets installés.