AI Capacity Building

## Table of contents

What is this project?.................................................................................................

Folders inside the repository………………………………………………………………………………..

Brief introduction to MLflow………………………………………………………………………………..

Information about the environment…………………………………………………………………….

Prerequisites………………………………………………………………………………………………………..

How to get started with VSCode(Optional)…………………………………………………………..

Opening MLflow User Interface……………………………………………………………………………

Frequently asked questions………………………………………………………………………………….

Resources…………………………………………………………………………………………………………….

List of Figures

Figure No. Figure Title Page No.

1. MLflow components
2. MLflow Tracking user interface
3. Docker Desktop containers section
4. Docker extension in VSCode
5. Dev Containers extension in VSCode

## What is this project?

This project is a virtual python development environment, a sandbox separating any python or packages you might have on your laptop with the projects in this environment so that these projects can run successfully.

It is made for the students to give them a few examples on machine learning and for the students to focus on developing their projects rather than them resolving packages dependencies.

## Folders in the repository

The repository can be found [here](https://github.com/AI-Capacity-Building/AI-Capacity-Building)

Every folder contains:

* The dataset used (if the dataset is not there, then the notebook downloads it inside the code)
* The notebook/ python file training the model

1. Accidents use case:

* This project aims to classify the accidents to 1 of 4 categories [1 - 4], 1 being less severe, 4 being the most severe
* The data is a csv file, consisting of 48 features and 1 label
* Dataset size is 819 MG and can be downloaded from [here](https://www.kaggle.com/code/deepakdeepu8978/how-severity-the-accidents-is/data)
* The classification model used is xgboost
* This project focuses a little bit more on the analysis of the data, feature importance, and visualization using matplotlib and seaborn

2. Images use case (Computer Vision):

* This project aims to classify whether an image contains an accident or no (binary classification).
* The data consists of 3 folders: train, validation, and test, each containing 2 folders: accidents images and non-accidents
* Dataset size is 250 MG and can be downloaded from [here](https://www.kaggle.com/datasets/ckay16/accident-detection-from-cctv-footage)
* The classification model used is a convolution neural network.

3. Smart Cities use case (Natural Language Processing):

* This project aims to classify Arabic reviews from hotels, books, movies whether they are positive, negative, or mixed
* The data is a tsv file (tab separated values file), consisting of the text and its label
* Dataset size is 51.8 MG and can be downloaded from [here](https://www.kaggle.com/datasets/abedkhooli/arabic-100k-reviews)
* The classification model used is support vector machines

4. Reinforcement Learning use case:

* Simulation of a self-driving cab using RL techniques in an OpenAI Gym environment.

5. Object Detection use case:

* A computer vision project detecting objects in images and videos using YOLO and OpenCV.

5. MLflow TensorFlow example (Computer Vision):

* This project is a simple example of how to log and track the parameters used in your TensorFlow model in MLflow, also it is an example of a dataset saved in Keras datasets.
* The dataset is MNIST, a set of images of handwritten numbers [0-9], 10 classes total
* The classification model used is a deep neural network made up of 3 dense layers

6. MLflow Torch example (Computer Vision):

* This project is a simple example of how to log and track the parameters used in your PyTorch model in MLflow, also it is an example of a dataset saved in Keras datasets.
* The dataset is MNIST, a set of images of handwritten numbers [0-9], 10 classes total

## Brief introduction to MLflow

Before talking about MLflow, we need to point out some differences between software engineering and machine learning

|  |  |
| --- | --- |
| Software Engineering | Machine Learning |
| Software engineering is about writing algorithms, that do a specific job, in form of functions that take specific inputs and return a specific output. | Machine learning is more about optimizing metrics (recall, ROC, r2 score, others) for performance, and trial and error is a huge part in this, whether it is trying different machine learning models, or trying the same model with different hyperparameters, or changing anything else. |

Where does MLflow come in all of this?

MLflow is an open-source platform that consists of 4 components

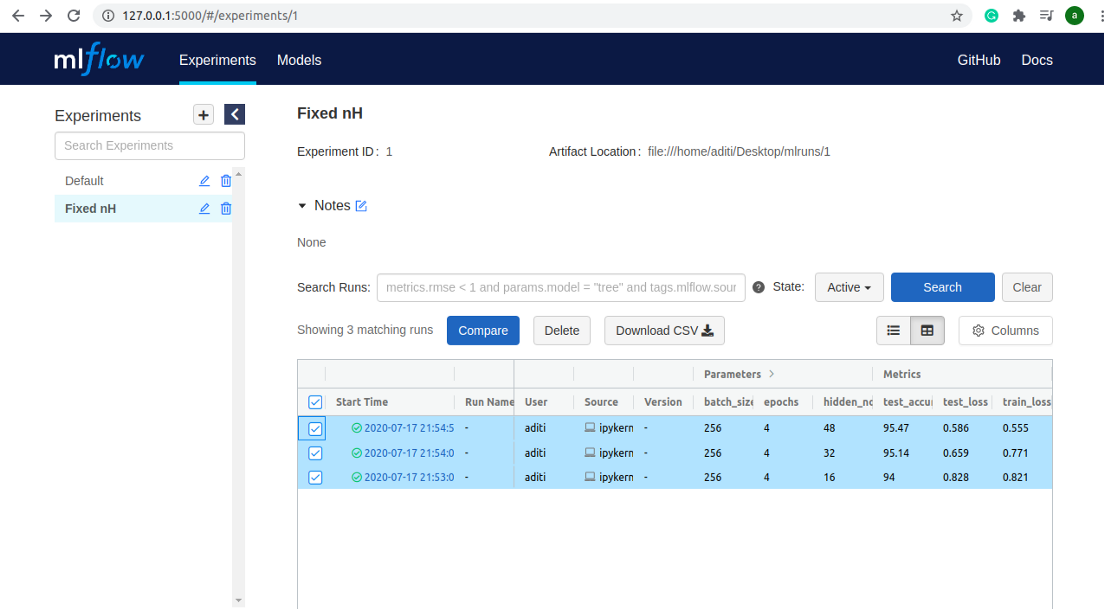
Diagram

Description automatically generatedFigure 1

One of the modules is MLflow Tracking which is used to track experiments to record and compare parameters (example: learning rate, regularization parameters, others) and results, which can be very useful.

In your code, you log the parameters you want to track in MLflow, and it has a separate user interface where you can view you runs.

MLflow is integrated in all the notebooks to help the students get comfortable with it.



## Figure 2

## Information about the environment

The python version used is 3.9.14-slim

Libraries installed inside the environment (sorted alphabetically) and their description

**Ipykernel**: this package enables developing in Jupyter notebooks

**Matplotlib**: a visualization library that creates static, animated, and interactive visualizations

**MLflow**: a library to organize, track and visualize your machine learning models

**NLTK**: a natural language processing toolbox, it gives us access to text processing libraries

**NumPy**: a library for mathematical functions, linear algebra and random number generators

**Pandas**: a library used for reading & writing data into forms that can be manipulated easily

**pytorch-lightning**: a library of high-performance that wraps around PyTorch that organizes PyTorch code. It makes deep learning experiments easier to read and reproduce.

**scikit-learn**: a library provides efficient tools for machine learning, it has built-in datasets and implemented machine learning models ready to be trained

**SciPy**: a library that has optimized implementation of many algorithms in low-level languages like C

**Seaborn**: a statistical data visualization library built on top of matplotlib

**TensorFlow**: open-source library used for numerical computation and machine learning model building

**Torch**: a library mostly used for deep learning using GPUs and CPUs

**torch audio**: an audio library for PyTorch, also provides GPU acceleration

**torch vision**: a computer vision library for PyTorch, has some datasets and pretrained models

**xgboost**: a library for the machine learning model with the same name (eXtreme Gradient Boosting)

**OpenCV:** Open-source library for computer vision and image processing, offering a wide range of tools and algorithms for tasks such as feature detection, object recognition, and image manipulation.

**Gym (OpenAI Gym)**: Toolkit by OpenAI for developing and comparing reinforcement learning algorithms, providing a set of environments for testing and benchmarking, including classic control tasks and Atari 2600 games.

## Prerequisites

1. You need to install docker desktop (link present in the resources section)
2. Get the docker image by either

* Pulling the docker image from the following link:
  + TensorFlow image from [here](https://hub.docker.com/r/aicapacitybuilding/tensorflow_env)
  + PyTorch image from [here](https://hub.docker.com/r/aicapacitybuilding/tensorflow_env)
* Building the Dockerfile to create the docker image by executing the following command: docker build -t ml\_env:2.0 .

1. Run the docker image by either

* Pressing the run button in docker desktop

Graphical user interface, text, application, email

Description automatically generated

Figure 3

* In your terminal execute: docker container run -p 5000:5000 ml\_env:2.0

## How to get started with VSCode (Optional)

1. Install VSCode from [here](https://code.visualstudio.com/Download)
2. Install these 2 extensions in VSCode: Docker, and Dev Containers

Graphical user interface, text

Description automatically generated

Figure 4

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5

1. If you haven't run the image yet, in your vscode terminal execute: docker container run -p 5000:5000 ml\_env:2.0
2. In VSCode, go to the docker icon present in the left
3. Under containers -> choose the running container (ml\_env:2.0) -> right click -> choose 'Attach Visual Studio Code'.
4. A newly opened VSCode will appear, install Jupyter & Python extensions (press install in container)
5. Press open folder
6. You can either choose a use case to run, or open root folder (folder containing all use cases) by just pressing ok
7. You can now use VSCode to develop inside the container
8. To run your code, open a terminal inside the example folder and write: python (python file name). Example: python mlflow\_torch\_example.py

## Opening MLflow User Interface

To open MLFlow user interface and view the runs output (make sure your code uses mlflow and logs the parameters you want to see)

1. Open a terminal in your project
2. Execute: mlflow ui --host 0.0.0.0
3. In your browser open: [http://0.0.0.0:5000](http://0.0.0.0:5000/) (mlflow uses port 5000 on the host)
4. If your experiment has a name (example : experiment\_name = "MNIST\_Classification"), it will create a folder with the same name in mlflow ui, open that folder to view the runs

## FAQ

* Do I have to use VSCode when I am developing inside the container?

No, you can use any IDE you like, just know how to integrate it with the container (check resources section)

* I am trying to run the docker image, but I keep getting errors?

Docker Desktop needs to be running whenever you are using the container

* Can I add new libraries in the environment?

Yes, but you need to make sure that:

* 1. The new library’s version doesn’t conflict with the other packages
  2. You need to rebuild the docker image

## Resources

* [What is a docker container?](https://www.docker.com/resources/what-container/)
* [Download docker desktop](https://www.docker.com/products/docker-desktop/) , for windows it is mandatory to install WSL2, [here](https://www.youtube.com/watch?v=OAtcxnNlSic) is how.
* [Docker tutorial for beginners](https://www.youtube.com/watch?v=3c-iBn73dDE)
* [MLflow documentation](https://mlflow.org/docs/latest/index.html)
* MLflow consists of 4 modules
  + [Tracking](https://www.youtube.com/watch?v=x3cxvsUFVZA)
  + [Projects & Models](https://www.youtube.com/watch?v=g5ibwiSH1uA)
  + [Model Registry](https://www.youtube.com/watch?v=AxYmj8ufKKY)
* Developing inside containers
* [Developing inside containers using PyCharm](https://www.jetbrains.com/help/pycharm/using-docker-as-a-remote-interpreter.html#prereq)
* [More details about debugging, sharing your environment, unit testing in containers (using VSCode and PyCharm)](https://nirradi.medium.com/vsc-vs-pycharm-developing-inside-docker-containers-4892c83d30e4)
* [6 metrics you need to optimize for performance in machine learning](https://dzone.com/articles/the-6-metrics-you-need-to-optimize-for-performance#:~:text=6%20Metrics%20You%20Need%20to%20Optimize%20for%20Performance,might%20not%20be%20the%20best%20for%20regression.%20)