**Collège de Bois de Boulogne**

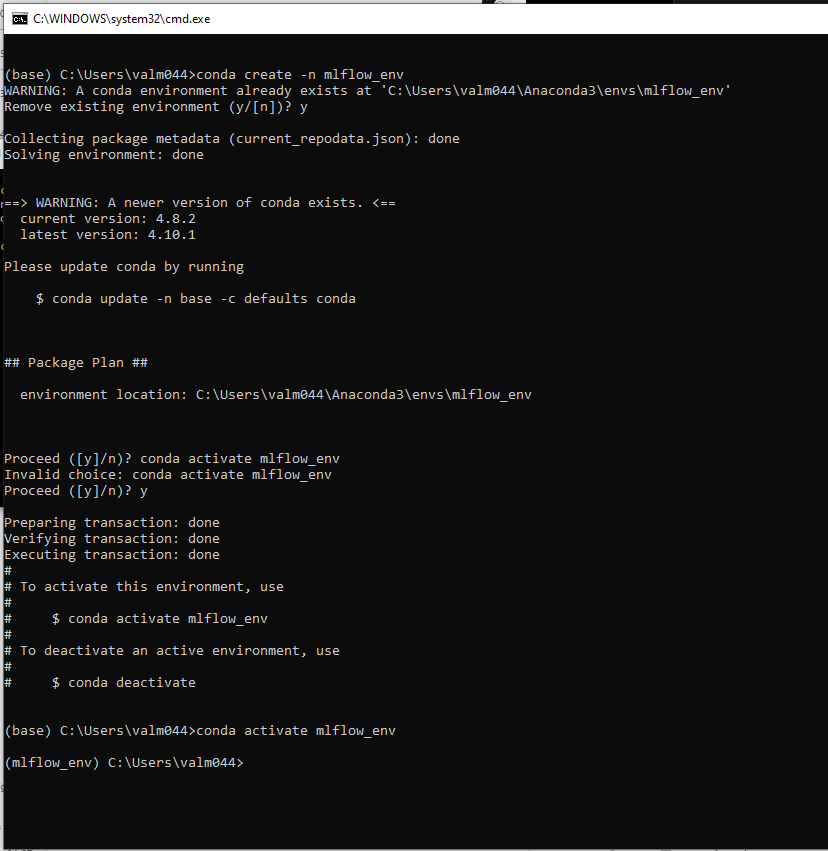
**Labo – Pratique mlflow**

**OBJECTIF**

1. Ouvrir une fenetre de commande et ouvrir un environnement conda

conda create -n mlflow\_env

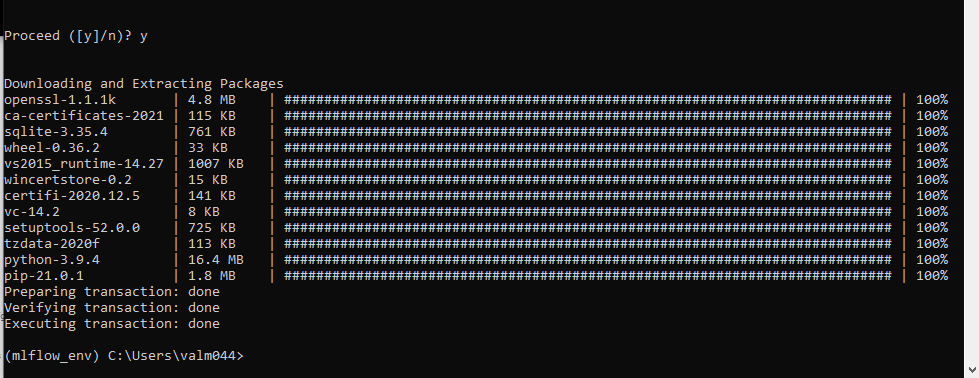
conda activate mlflow\_env



staller la librairie mlflow

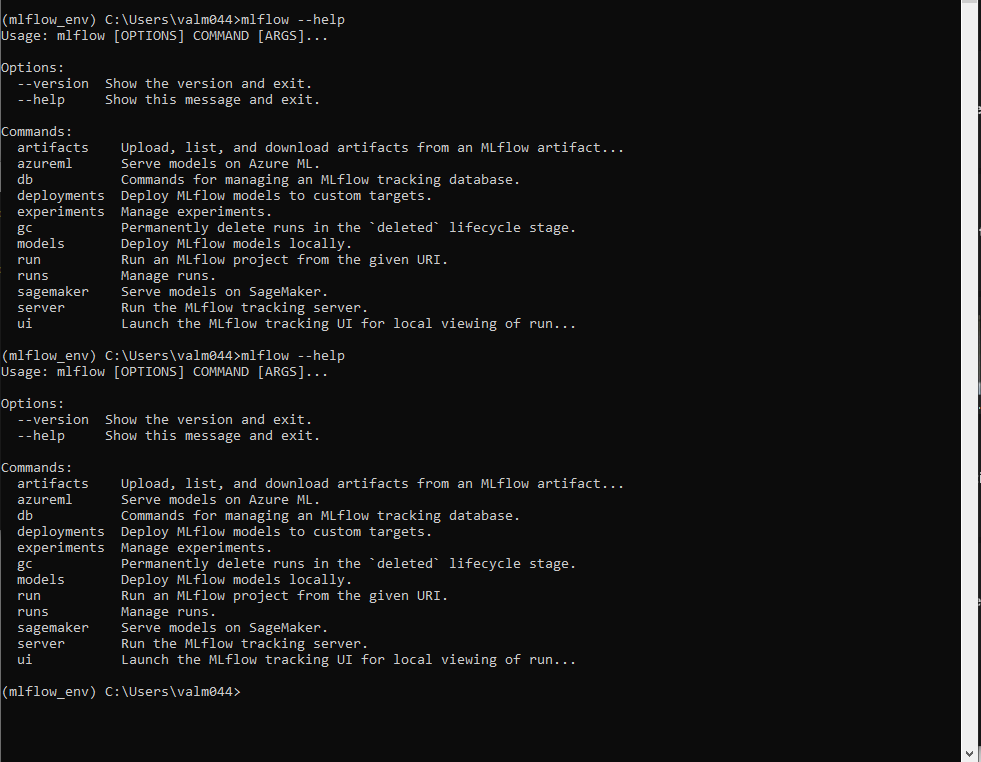
conda install python

pip install mlflow



1. Vérifier que MLflow est bien installé

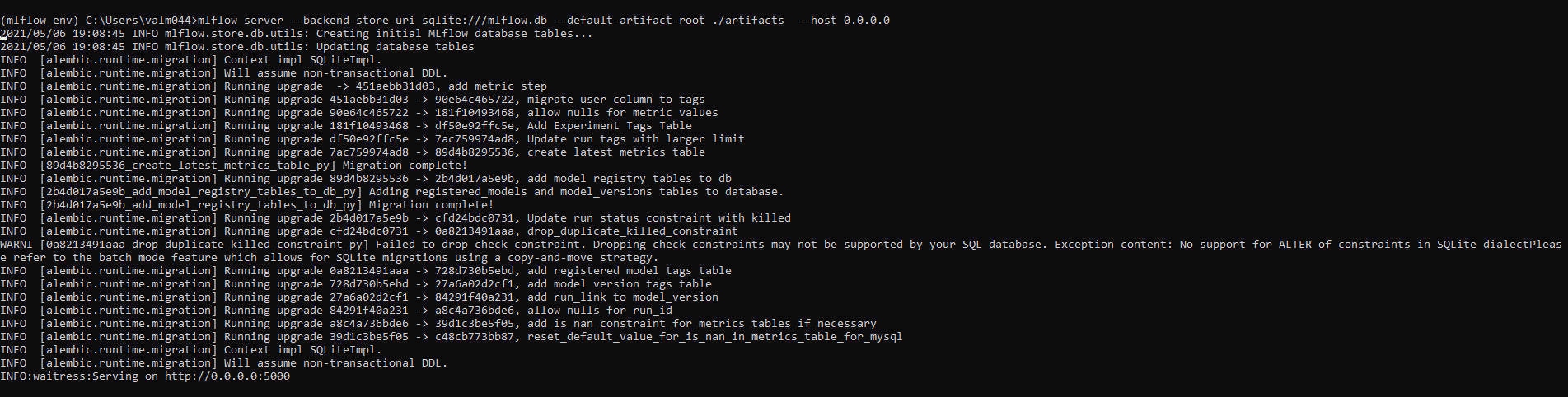
mlflow --help

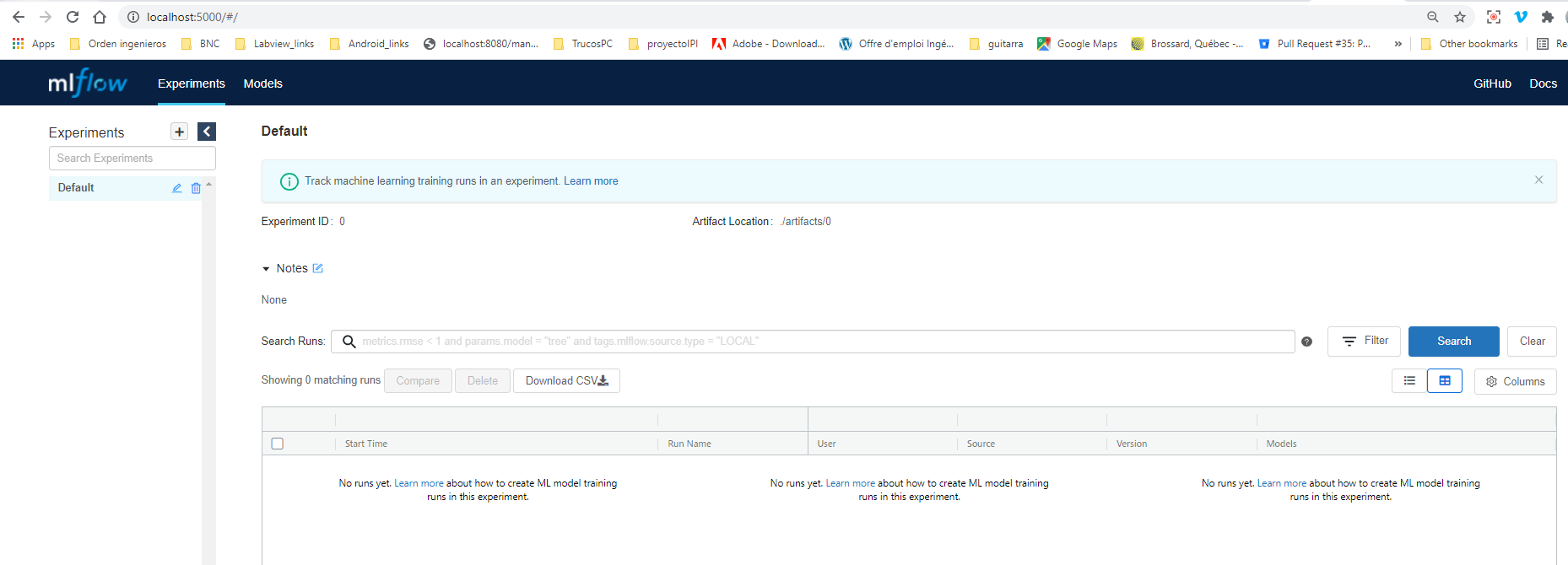


1. Exécuter l’interface UI de mlflow

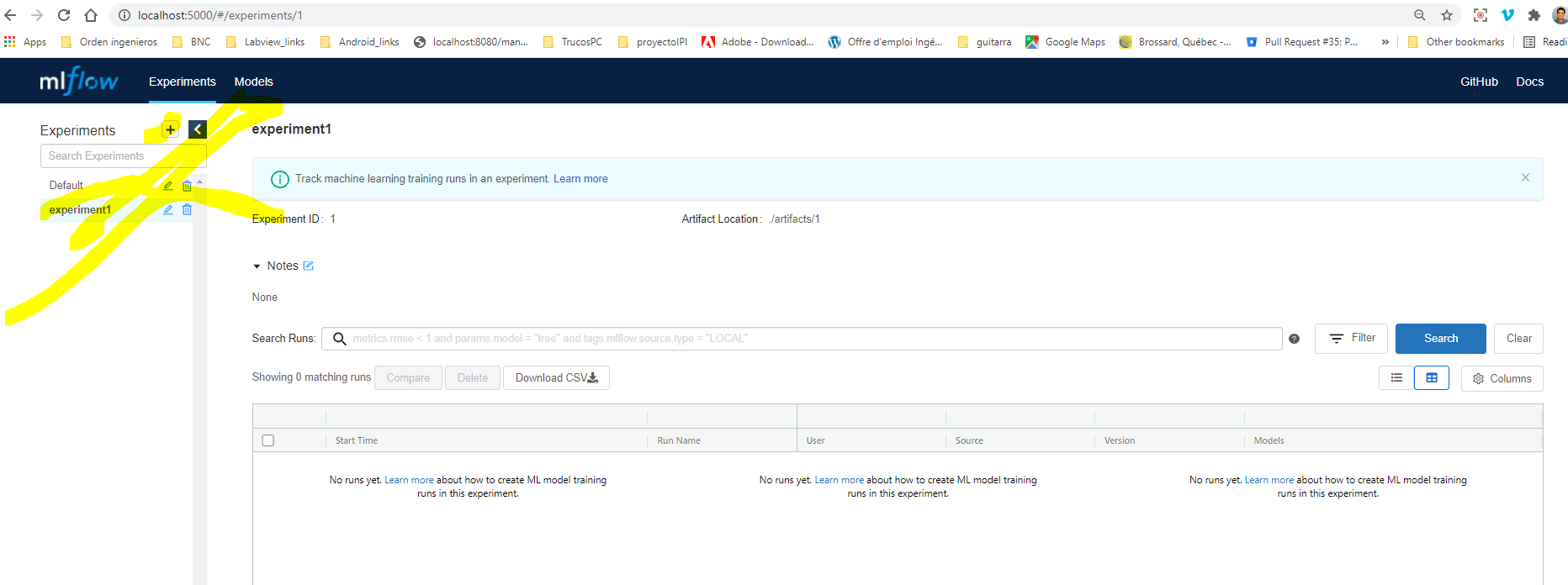
**mlflow server --backend-store-uri sqlite:///mlflow.db --default-artifact-root ./artifacts --host 0.0.0.0**

Note: à vérifier (suggestion de Zakaria)





Create experiment :



1. Ouvrir un jupyter notebook
2. Utiliser le code suivant :

# The data set used in this example is from http://archive.ics.uci.edu/ml/datasets/Wine+Quality

# P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.

# Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

#! pip install --user mlflow

import os

import warnings

import sys

import pandas as pd

import numpy as np

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import ElasticNet

import mlflow

import mlflow.sklearn

import logging

logging.basicConfig(level=logging.WARN)

logger = logging.getLogger(\_\_name\_\_)

def eval\_metrics(actual, pred):

rmse = np.sqrt(mean\_squared\_error(actual, pred))

mae = mean\_absolute\_error(actual, pred)

r2 = r2\_score(actual, pred)

return rmse, mae, r2

if \_\_name\_\_ == "\_\_main\_\_":

warnings.filterwarnings("ignore")

np.random.seed(40)

# Read the wine-quality csv file from the URL

csv\_url =\

'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'

try:

data = pd.read\_csv(csv\_url, sep=';')

except Exception as e:

logger.exception(

"Unable to download training & test CSV, check your internet connection. Error: %s", e)

# Split the data into training and test sets. (0.75, 0.25) split.

train, test = train\_test\_split(data)

# The predicted column is "quality" which is a scalar from [3, 9]

train\_x = train.drop(["quality"], axis=1)

test\_x = test.drop(["quality"], axis=1)

train\_y = train[["quality"]]

test\_y = test[["quality"]]

#alpha = float(sys.argv[1]) if len(sys.argv) > 1 else 0.5

#l1\_ratio = float(sys.argv[2]) if len(sys.argv) > 2 else 0.5

alpha = 0.67#à changer

l1\_ratio = 0.3#à changer

#Note de Zakaria

**mlflow.set\_experiment(experiment\_name='experiment1')**

**mlflow.set\_tracking\_uri("http://localhost:5000")**

with mlflow.start\_run():

lr = ElasticNet(alpha=alpha, l1\_ratio=l1\_ratio, random\_state=42)

lr.fit(train\_x, train\_y)

predicted\_qualities = lr.predict(test\_x)

(rmse, mae, r2) = eval\_metrics(test\_y, predicted\_qualities)

print("Elasticnet model (alpha=%f, l1\_ratio=%f):" % (alpha, l1\_ratio))

print(" RMSE: %s" % rmse)

print(" MAE: %s" % mae)

print(" R2: %s" % r2)

mlflow.log\_param("alpha", alpha)

mlflow.log\_param("l1\_ratio", l1\_ratio)

mlflow.log\_metric("rmse", rmse)

mlflow.log\_metric("r2", r2)

mlflow.log\_metric("mae", mae)

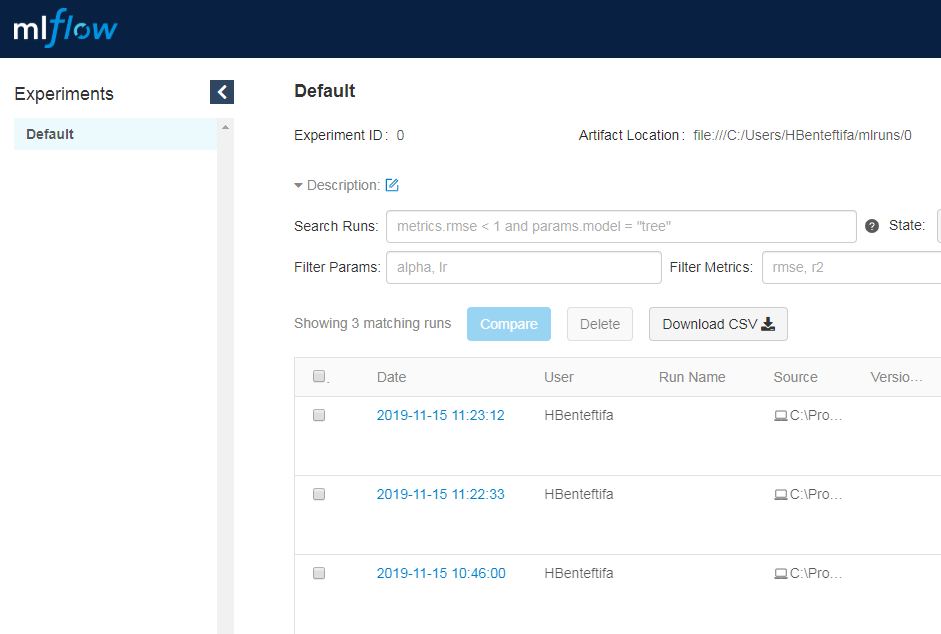
mlflow.sklearn.log\_model(lr, "model")

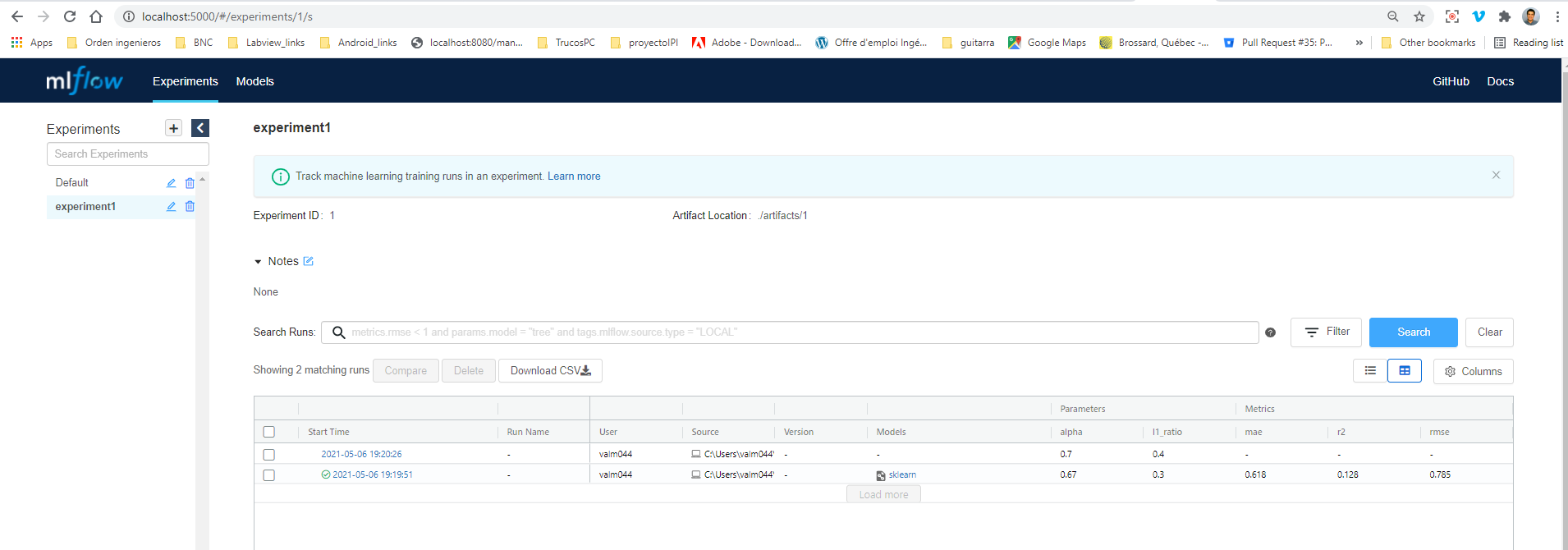
1. Procéder à son exécution (changer les valeurs indiquées dans le code)

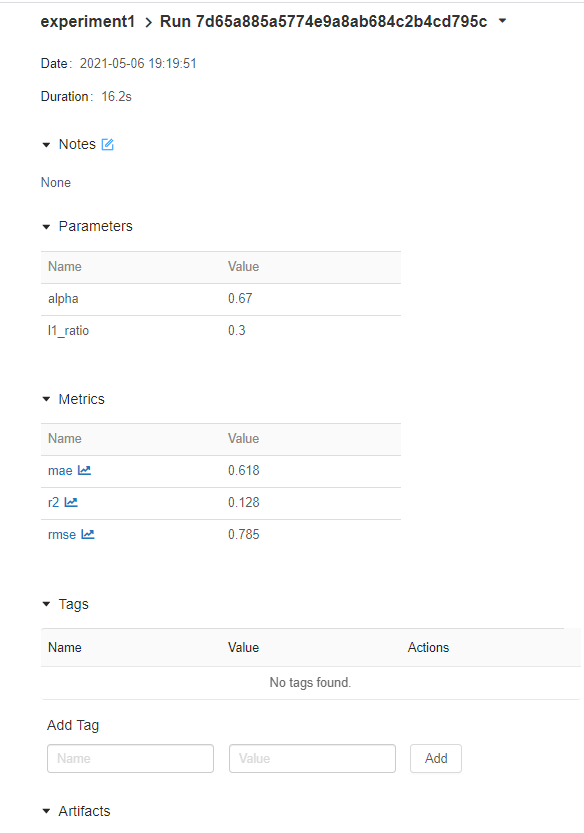
alpha = 0.67

l1\_ratio = 0.3

1. Vérifier sur l’interface MLflow que vous avez un suivi de vos exécutions







**Références**

<https://thegurus.tech/posts/2019/06/mlflow-production-setup/>

<https://mlflow.org/docs/latest/tutorial.html>