EPITA - International Programs M2
AI Project Methodology
2021-2022

Lab MLFlow

MLFlow Documentation

- MLFlow Tracking**:** https://mlflow.org/docs/latest/tracking.html
- MLFlow Projects: https://mlflow.org/docs/latest/projects.html
- MLFlow Models: https://mlflow.org/docs/latest/models.html
- MLFlow Model Registery : https://mlflow.org/docs/latest/model-registry.html

GOALS

Use MLflow to:

- Train a linear regression model and track its parameters
- Package the code that trains the model in a reusable and reproducible model format
- Deploy the model into a simple HTTP server that will enable you to score predictions

REQUIREMENTS

Python 3.6 or later installed (you can use anaconda environment for example)

Linux is preferred for this Lab.

You may need to have git installed (https://git-scm.com/downloads)

MLFLOW INSTALLATION

- Install MLflow :

  pip install mlflow
- Install scikit-learn :

  pip install scikit-learn
- You can also Install MLflow with extra dependencies, including scikit-learn :

  pip install mlflow[extras]
- Install conda :

  pip install conda
- Download or Clone the MLflow repository and save it in your local working directory at :

  https://github.com/mlflow/mlflow

1/Example Project

Unzip the mlflow-master.zip and cd into the “examples\sklearn_elasticnet_wine” directory to view the example project (we’ll use this working directory for running this Lab).

Dataset

This example project uses a dataset to predict the quality of wine based on quantitative features like the wine’s “fixed acidity”, “pH”, “residual sugar”, and so on.

The dataset is from UCI’s machine learning repository

Training the Model

This example uses the familiar pandas, numpy, and sklearn APIs to create a simple machine learning model.

First, train a linear regression model that takes two hyperparameters: alpha and l1_ratio.

The code is located at examples/sklearn_elasticnet_wine/train.py

- Run the example with default hyperparameters “alpha” and “l1_ratio” as follows:

```
python train.py
```

You are expected to get this result :

```
Elasticnet model (alpha=0.500000, l1_ratio=0.500000):
  RMSE: 0.7931640229276851
  MAE: 0.6271946374319587
  R2: 0.10862644997792614
```

- Try out some other values for “alpha” and “l1_ratio” by passing them as arguments to train.py:

```
python train.py <alpha> <l1_ratio>
```

Ex : python train.py 1 0.5

MLFlow Tracking

The MLflow tracking APIs log information about each training run, like the hyperparameters alpha and l1_ratio, used to train the model and metrics, like the root mean square error, used to evaluate the model. The example also serializes the model in a format that MLflow knows how to deploy.

Usage of MLFlow in the code (train.py)

```
import mlflow
import mlflow.sklearn
```

.....

```
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)
```

Each time you run the example, MLflow logs information about your experiment runs in the local directory **mlruns.**

- Cd to **mlruns** directory and browse throw the subdirectories

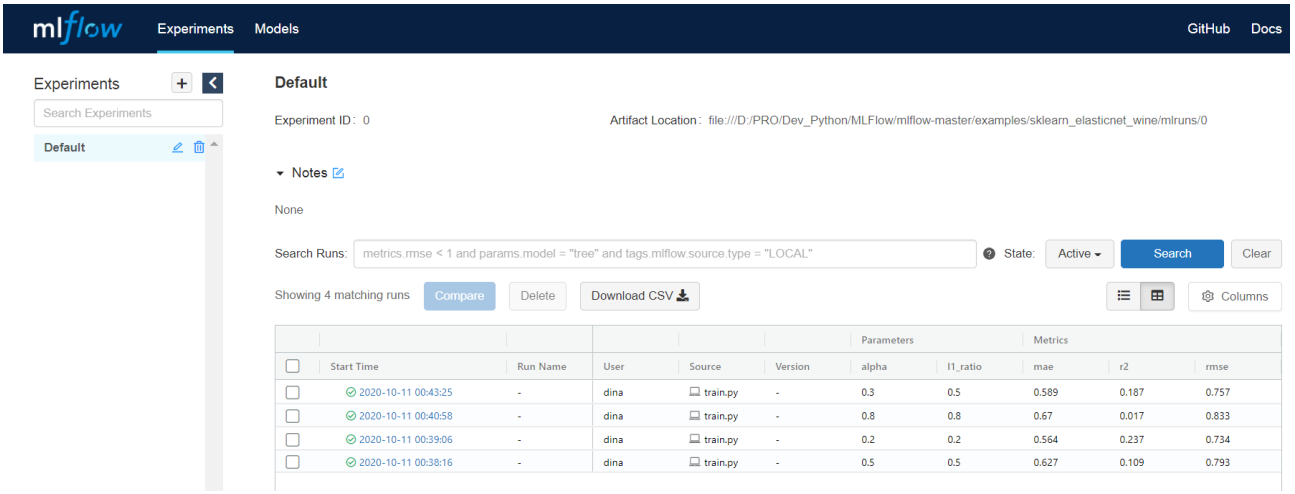
Comparing the models with MLflow UI

Use the **MLflow UI** to compare the models that you have produced.

- In the same working directory as the one that contains the **mlruns** (examples/sklearn_elasticnet_wine) run:

  ```
  mlflow ui
  ```

- View it at http://localhost:5000



On this page, you can see the list of your experiment runs with metrics, you can use them to compare the models.

- Use the search feature to quickly filter out many models.
  For example, the query **metrics.rmse < 0.8** returns all the models with root mean squared error less than 0.8. For more complex manipulations, you can download this table as a CSV and use your favorite data munging software to analyze it.
- Download CSV and take a look to it.

MLFlow Projects

Packaging Training Code in a Conda Environment

Now that you have your training code, you can package it so that other data scientists can easily reuse the model, or so that you can run the training remotely (for example on Databricks), by using **MLFlow Projects** conventions to specify the dependencies and entry points to your code.

- The **MLproject** file (in example repo sklearn_elasticnet_wine) specifies that the project has the dependencies located in a Conda environment file called **conda.yaml** and has one entry point that takes two parameters : alpha and l1_ratio.

```
# sklearn_elasticnet_wine/MLproject

name: tutorial

conda_env: conda.yaml

entry_points:
  main:
    parameters:
      alpha: float
      l1_ratio: {type: float, default: 0.1}
    command: "python train.py {alpha} {l1_ratio}"
```

- The **conda.yaml** file lists the dependencies:

```
# sklearn_elasticnet_wine/conda.yaml

name: tutorial
channels:
  - defaults
dependencies:
  - numpy=1.14.3
  - pandas=0.22.0
  - scikit-learn=0.19.1
  - pip:
    - mlflow
```

- To run this project, invoke in the "\mlflow-master" or in current working directory :
  `mlflow run examples\sklearn_elasticnet_wine -P alpha=0.42`

```
mlflow run . -P alpha=0.42
```

- After running this command, MLflow runs your training code in a new Conda environment with the dependencies specified in `conda.yaml`.

```
PS D:\PRO\Dev_Python\MLFlow\mlflow-master> mlflow run examples\sklearn_elasticnet_wine -P alpha=0.42
c:\programdata\anaconda3\lib\site-packages\IPython\lib\pretty.py:91: DeprecationWarning: IPython.utils.signatures backport for Python
 2 is deprecated in IPython 6, which only supports Python 3
  from IPython.utils.signatures import signature
2020/10/12 00:39:23 INFO mlflow.projects.utils: === Created directory C:\Users\dina\AppData\Local\Temp\tmpirz69l0h for downloading re
mote URIs passed to arguments of type 'path' ===
2020/10/12 00:39:23 INFO mlflow.projects.backend.local: === Running command 'conda activate mlflow-6284a367a61b51ccdf445333a216776597
fb4efc && python train.py 0.42 0.1' in run with ID '0113a12367cf4ee0bd27f5f3edb6d44a' ===
Elasticnet model (alpha=0.420000, l1_ratio=0.100000):
  RMSE: 0.7420620899060748
  MAE: 0.5722846717246248
  R2: 0.2197851365155024 7
2020/10/12 00:39:29 INFO mlflow.projects: === Run (ID '0113a12367cf4ee0bd27f5f3edb6d44a') succeeded ===
```

- If the repository has an `MLproject` file in the root you can also run a project directly from GitHub. This project is duplicated in the https://github.com/mlflow/mlflow-example repository which you can run with (git must be installed and properly configured) :

```
mlflow run https://github.com/mlflow/mlflow-example.git -P alpha=5.
```

MLFLOW MODELS

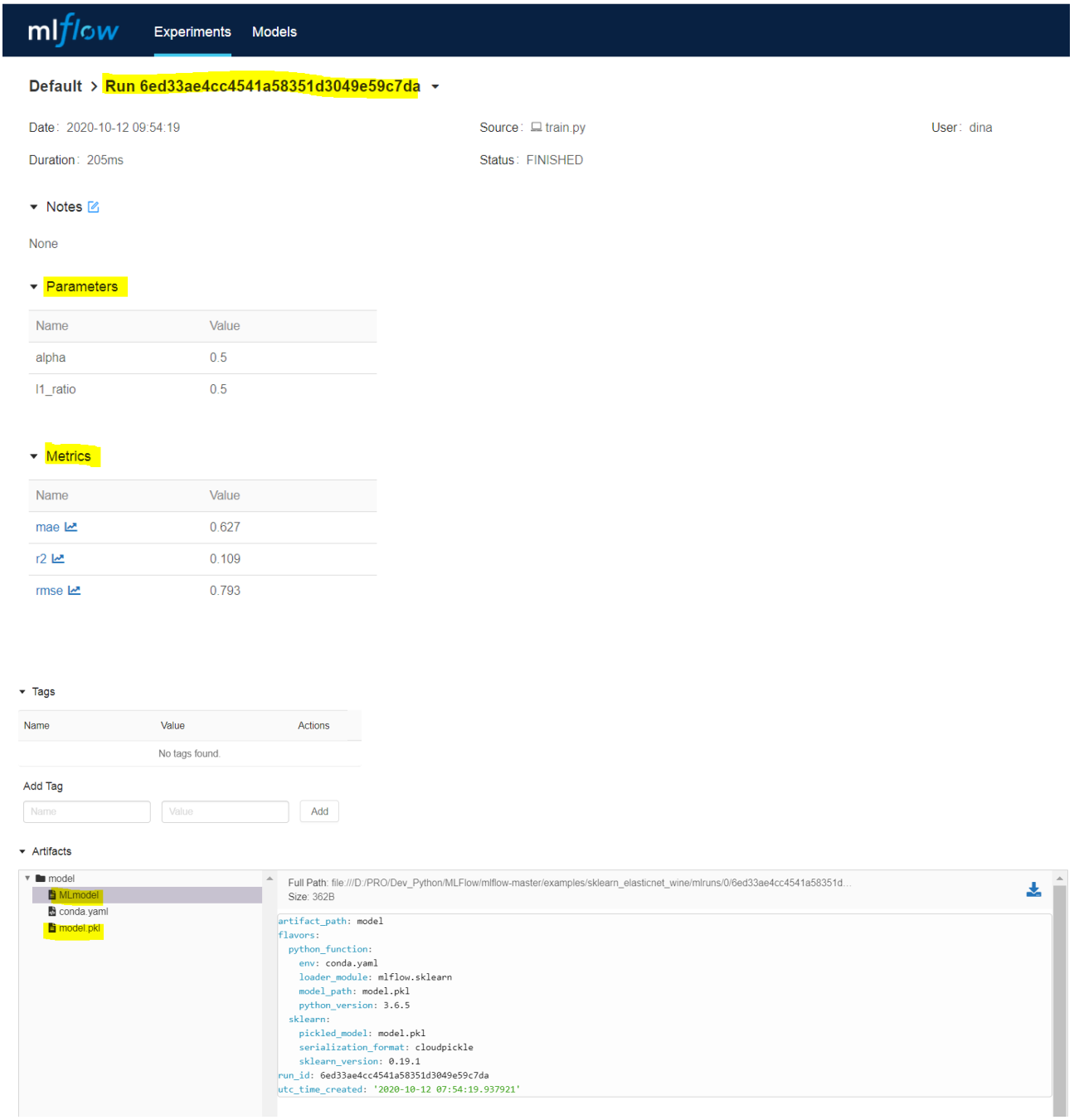
Serving the Model

Now that you have packaged your model using the MLFlow Projects convention, it's possible to deploy the model using **MLflow Models**. An MLflow Model is a standard format for packaging machine learning models that can be used in a variety of downstream tools — for example, real-time serving through a REST API or batch inference on Apache Spark.

In the example training code (train.py), after training the linear regression model, a function in MLflow saved the model as an artifact within the run **:**

```
mlflow.sklearn.log_model(lr, "model")
```

- To view this artifact, you can use the UI again (http://localhost:5000). When you click a date in the list of experiment runs you'll see this page.



- We can see that the call to `mlflow.sklearn.log_model` produced two files :

  – The first file "MLmodel" is a metadata file that tells MLflow how to load the model.
  – The second file "model.pkl" is a serialized version of the linear regression model that you trained.

- These 2 files can be found in the "mlruns" repository corresponding to the experimentation (replace the path with your model's path):

`examples\sklearn_elasticnet_wine\mlruns\0\6ed33ae4cc4541a58351d3049e59c7da`

Ce PC ▸ Data (D:) ▸ PRO ▸ Dev_Python ▸ MLFlow ▸ mlflow-master ▸ examples ▸ sklearn_elasticnet_wine ▸ mlruns ▸ 0

Nom	Modifié le	Type	Taille
meta.yaml	11/10/2020 00:38	Fichier YAML	1 Ko
deb8bbb655314a8d9db7d7e85d2fbb98	12/10/2020 10:05	Dossier de fichiers	
ce3ee5125d63497aa157be37c3bfdc21	12/10/2020 10:01	Dossier de fichiers	
8186fd8fe6d74f89a705c94b15cf1099	12/10/2020 09:57	Dossier de fichiers	
6ed33ae4cc4541a58351d3049e59c7da	12/10/2020 09:54	Dossier de fichiers	
5d8c25ac886a41658c4e5dc302940846	12/10/2020 00:07	Dossier de fichiers	
ee841b4111824e728a18e9de715da5e5	11/10/2020 22:30	Dossier de fichiers	
5890e6f0c20b4a39ae774b03cf4d2664	11/10/2020 22:30	Dossier de fichiers	
39b85376cfdb47c58ecac698af867703	11/10/2020 00:43	Dossier de fichiers	
040499fc36b34274b41d35905c1b49ed	11/10/2020 00:40	Dossier de fichiers	
a9dd7b32841341c882bd1b665f9464dc	11/10/2020 00:39	Dossier de fichiers	
6915ca5d6a8548528a6cdff7a37da99a	11/10/2020 00:38	Dossier de fichiers	

Ce PC ▸ Data (D:) ▸ PRO ▸ Dev_Python ▸ MLFlow ▸ mlflow-master ▸ examples ▸ sklearn_elasticnet_wine ▸ mlruns ▸ 0 ▸ 6ed33ae4cc4541a58351d3049e59c7da

Nom	Modifié le	Type	Taille
artifacts	12/10/2020 09:54	Dossier de fichiers	
metrics	12/10/2020 09:54	Dossier de fichiers	
params	12/10/2020 09:54	Dossier de fichiers	
tags	12/10/2020 09:54	Dossier de fichiers	
meta.yaml	12/10/2020 09:54	Fichier YAML	1 Ko

« PRO ▸ Dev_Python ▸ MLFlow ▸ mlflow-master ▸ examples ▸ sklearn_elasticnet_wine ▸ mlruns ▸ 0 ▸ 6ed33ae4cc4541a58351d3049e59c7da ▸ artifacts ▸ model

Nom	Modifié le	Type	Taille
conda.yaml	12/10/2020 09:54	Fichier YAML	1 Ko
MLmodel	12/10/2020 09:54	Fichier	1 Ko
model.pkl	12/10/2020 09:54	Fichier PKL	1 Ko

You can use this MLmodel format with MLflow to deploy a local REST server that can serve predictions.

- To deploy the server, run (replace the path with your model's path):

```
mlflow models serve -m
mlruns\0\6ed33ae4cc4541a58351d3049e59c7da\artifacts\model -p 1234
```

or (if –m don't work)

```
mlflow models serve --model-uri
mlruns\0\6ed33ae4cc4541a58351d3049e59c7da\artifacts\model -p 1234
```

Serving on **http://localhost:1234/**

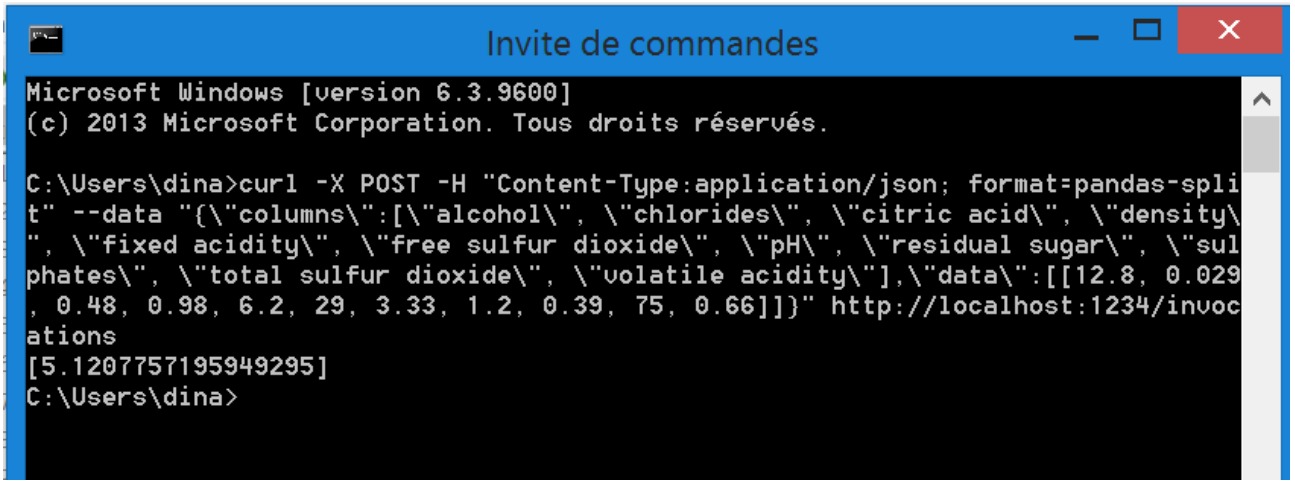
Once you have deployed the server, you can pass it some sample data and see the predictions. The following example uses curl to send a JSON-serialized pandas DataFrame with the split orientation to the model server :

WARNING : this command works in Bash environment but don't work in PowerShell, use command prompt on windows .

```
curl -X POST -H "Content-Type:application/json; format=pandas-split" --data
'{"columns":["alcohol", "chlorides", "citric acid", "density", "fixed acidity", "free
sulfur dioxide", "pH", "residual sugar", "sulphates", "total sulfur dioxide", "volatile
acidity"],"data":[[12.8, 0.029, 0.48, 0.98, 6.2, 29, 3.33, 1.2, 0.39, 75, 0.66]]}'
http://127.0.0.1:1234/invocations
```

From windows Command prompt :

curl -X POST -H "Content-Type:application/json; format=pandas-split" --data "{\"columns\":[\"alcohol\", \"chlorides\", \"citric acid\", \"density\", \"fixed acidity\", \"free sulfur dioxide\", \"pH\", \"residual sugar\", \"sulphates\", \"total sulfur dioxide\", \"volatile acidity\"],\"data\":[[12.8, 0.029, 0.48, 0.98, 6.2, 29, 3.33, 1.2, 0.39, 75, 0.66]]}" http://localhost:1234/invocations



For more information about the input data formats accepted by the model server, see the MLflow deployment tools documentation. https://mlflow.org/docs/latest/models.html#local-model-deployment

2/More Examples

Based on the example above, test MLflow features on another ML project (H20, Keras, XGBoost, PyToch TensorFlow….).

You can find some example projects in **mlflow-master\example**