# MLflow and Hydra Notes

## Introduction

This document is primarily concerned with giving me a template to run these commands. One that I can apply to new situations. We are going to use the NYC\_Predict\_Rental\_Prices as the example.

### Steps

We are going to run this in steps to start with.

#### download

There is no directory for this. This is a preliminary step. The code was already set up for me. To run this:  
mlflow run . -P steps=download

The code in main.py looks like this.

##### if "download" in active\_steps:

##### # Download file and load in W&B

##### \_ = mlflow.run(

##### f"{config['main']['components\_repository']}/get\_data",

##### "main",

##### version='main',

##### parameters={

##### "sample": config["etl"]["sample"],

##### "artifact\_name": "sample.csv",

##### "artifact\_type": "raw\_data",

##### "artifact\_description": "Raw file as downloaded"

##### },

##### )

This goes to config.yaml and puts together the path to get sample1.csv out of github. This command uploads sample1 as sample.csv:latest to wandb

Now that sample.csv is on wandb, you next download it and look at it in a Jupyter Notebook using this command.

##### mlflow run src/eda

You need to be in the parent directory to do this. It needs to look at the MLproject file at the top and then go down 2 levels to eda and look at that one. I found this would start a very poor Jupyter environment. I did not like working in it and it seemed to have problems saving the code to wandb. I just gave up and saved the code as an artifact.

#### EDA

We are doing a preliminary look at the data with this step. This is in the directory eda. There are no further Hydra or MLflow commands here.

#### basic\_cleaning

You need to be in the root directory of this project. Then you can run cookiecutter.

##### cookiecutter cookie-mlflow-step -o src

##### step\_name [step\_name]: basic\_cleaning

##### script\_name [run.py]: run.py

##### job\_type [my\_step]: basic\_cleaning

##### short\_description [My step]: A very basic data cleaning

##### long\_description [An example of a step using MLflow and Weights & Biases]: Download from W&B the raw dataset and apply some basic data cleaning, exporting the result to a new artifact

##### parameters [parameter1,parameter2]: input\_artifact,output\_artifact,output\_type,output\_description,min\_price,max\_price

The MLflow command for this RAN from the root directory is:

##### mlflow run . -P steps="basic\_cleaning"

There was some work to get all of the run.py, main.py, conda.yaml (2 files basic\_cleaning and root directory), MLflow file (local to basic\_cleaning) in shape. Just follow those as the examples.

#### data\_check

So, this appeared to be set up for me. I had to make sure that the config.yaml file worked with the MLproject file in the data\_check directory. I also had to update the main.py file. The big issue turned out to be getting all of the names correct thru the various files. One odd thing is that there are 2 calling functions for the tests. One tests the current sample dataset. The other will test a future one. This is a little puzzling but understandable. The notes do warn you about this but … still strange. To run this:

##### mlflow run . -P steps="data\_check"

#### train\_val\_test\_split

Well I did a poor job of this. So, main.py you should be using config. EVERYWHERE else, no config. Just use the parameter name. All of the substitution is handled thru Hydra. I also changed the name of this step. I probably should not have but I did. I want to have a directory for every function. The steps they appear to be given are not exactly that. That is the name of the step does not correspond one to one to the function name. I find that very confusing. It is used to be split\_data. I also decided to upload 3 datasets not just 2. They were train, test, and validation. The notes only wanted trainval and test. Weird. Not sure why.

##### mlflow run . -P steps="train\_val\_test\_split"

Don’t use cookie cutter anymore. Just copy the files over from the last directory used.

#### train\_random\_forest

I got this working without that much difficulty.

##### mlflow run . -P steps=" train\_random\_forest"

#### all

I then ran all of them together and had some problems. I did get it working though.

##### mlflow run . -P steps=" all"