UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Build an ML Pipeline for Short-term Rental Prices in NYC

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Udacity Student

You have done an amazing job in this project and it will open a huge door in your future work. MLOps is a very important topic on every field of machine learning. If you want models into production and reliable pipelines, you are on the right path!

As a reference, I would like to share some articles and resources to boost your learning:

MLOps: Continuous delivery and automation pipelines in machine learning

MLOps: What It Is, Why it Matters, and How To Implement It (from a Data Scientist Perspective)

MLOps Core

10 Amazing MLOps Learning Resources

MLOps-Reducing the technical debt of Machine Learning

I really hope you enjoy the project as we mentors did and we are looking forward your next step!

W&B Set-Up

Your W&B project nyc_airbnb should be made public, so that your reviewer can access it. This is needed so the reviewer can check that the W&B steps have been executed successfully.

Make sure the link to your W&B project, as well as your Github repository (i.e. two links), are included in a README file or given to the reviewer in the "Submission Details" box you can use when initiating the submission process.

W&B Set-Up

W&B Project and Github

Great! 😉

Your Github and W&B project are public!

Exploratory Data Analysis

There is a sample.csv artifact in W&B.

The pipeline has been run to get a sample of the data, which has been uploaded to W&B.

Exploratory Data Analysis

Sample Dataset Artifact

Great job here as well! 😉

When running your pipeline, it is possible to get the sample of the data, which has been uploaded to W&B.

There is a notebook called EDA in the students' repository (most probably in the src/eda directory).

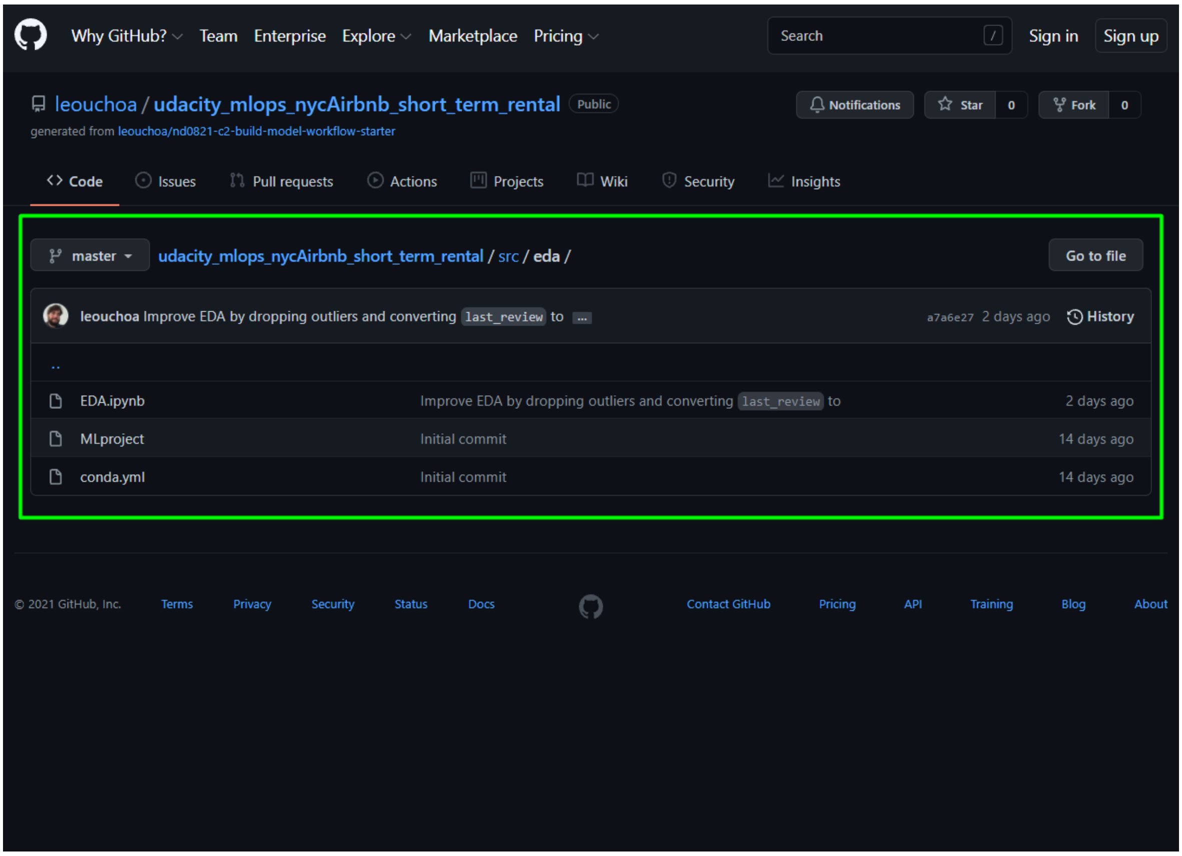
The EDA notebook contains a properly formatted Jupyter notebook with comments and markdown cells.

Exploratory Data Analysis

EDA Notebook

Great! 😄

Your Github repository contains a properly formatted Jupyter notebook.



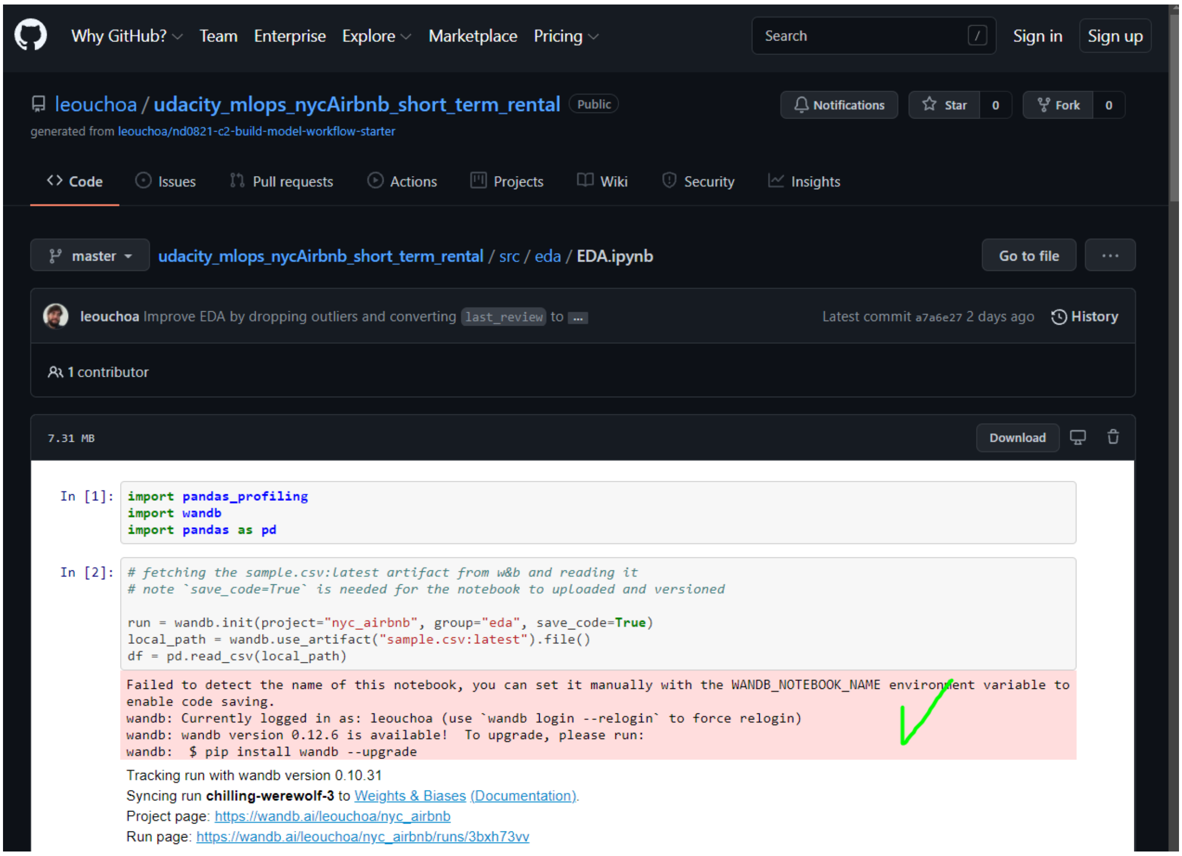
At the beginning of the notebook, fetch the sample.csv artifact from W&B.

Exploratory Data Analysis

W&B Fetching Artifact from W&B

Perfect! 😉

At the beginning of your notebook, you fetch the *sample.csv* artifact from your W&B.



The data is clean at the end of the notebook. Note that there will still be some missing entries, because we are not imputing missing values.

Properly implemented the checks suggested in the `notes.md` file.

Exploratory Data Analysis

Data Cleaning

At the end of the notebook, we can verify that the data is clean and without any other issue.

```
In [5]: profile = pandas_profiling.ProfileReport(df, explorative=True)
        profile.to_widgets()

        Summarize dataset: 100%|██████████| 29/29 [00:29<00:00,  1.01s/it, Completed]
        Generate report structure: 100%|██████████| 1/1 [00:10<00:00, 10.92s/it]
        VBox(children=(Tab(children=(Tab(children=(GridBox(children=(VBox(children=(GridspecLayout(children=(HTML(valu…

In [6]: profile

        Render HTML: 100%|██████████| 1/1 [00:06<00:00,  6.66s/it]

Out[6]:

In [7]: min_price = 10
        max_price = 350
        idx = df['price'].between(min_price, max_price)
        df = df[idx].copy()
        # Convert last_review to datetime
        df['last_review'] = pd.to_datetime(df['last_review'])
        df.info()

        <class 'pandas.core.frame.DataFrame'>
        Int64Index: 19001 entries, 0 to 19999
        Data columns (total 16 columns):
         #   Column                          Non-Null Count  Dtype
        ---  ------                          --------------  -----
         0   id                              19001 non-null  int64
         1   name                            18994 non-null  object
         2   host_id                         19001 non-null  int64
         3   host_name                       18993 non-null  object
         4   neighbourhood_group             19001 non-null  object
         5   neighbourhood                   19001 non-null  object
         6   latitude                        19001 non-null  float64
         7   longitude                       19001 non-null  float64
         8   room_type                       19001 non-null  object
         9   price                           19001 non-null  int64
         10  minimum_nights                  19001 non-null  int64
         11  number_of_reviews               19001 non-null  int64
         12  last_review                     15243 non-null  datetime64[ns]
         13  reviews_per_month               15243 non-null  float64
         14  calculated_host_listings_count  19001 non-null  int64
         15  availability_365                19001 non-null  int64
        dtypes: datetime64[ns](1), float64(3), int64(7), object(5)
        memory usage: 2.5+ MB

In [8]: run.finish()

        Waiting for W&B process to finish, PID 5414
        Program ended successfully
```

Data Cleaning

There is a new “basic_cleaning” step in the Github repository (under the src directory).

The basic_cleaning step respects the MLFlow structure: a conda.yml, a MLproject and a python script. It has the parameters input_artifact, output_name, output_type, output_description, min_price and max_price.

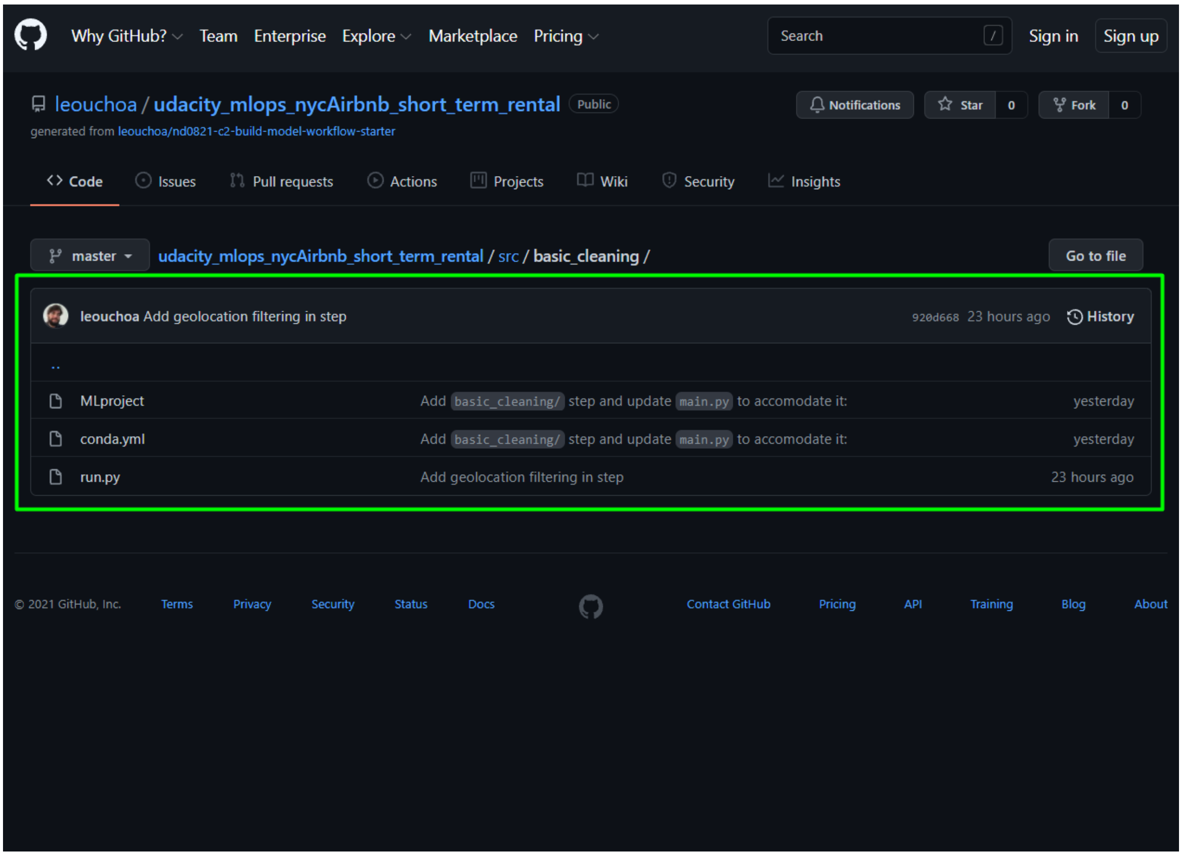
Data Cleaning

The *“basic_cleaning”* Step

Perfect! 😉

Your *“basic_cleaning”* step respects the MLFlow structure:

✔ - conda.yml
✔ - MLproject
✔ - python script

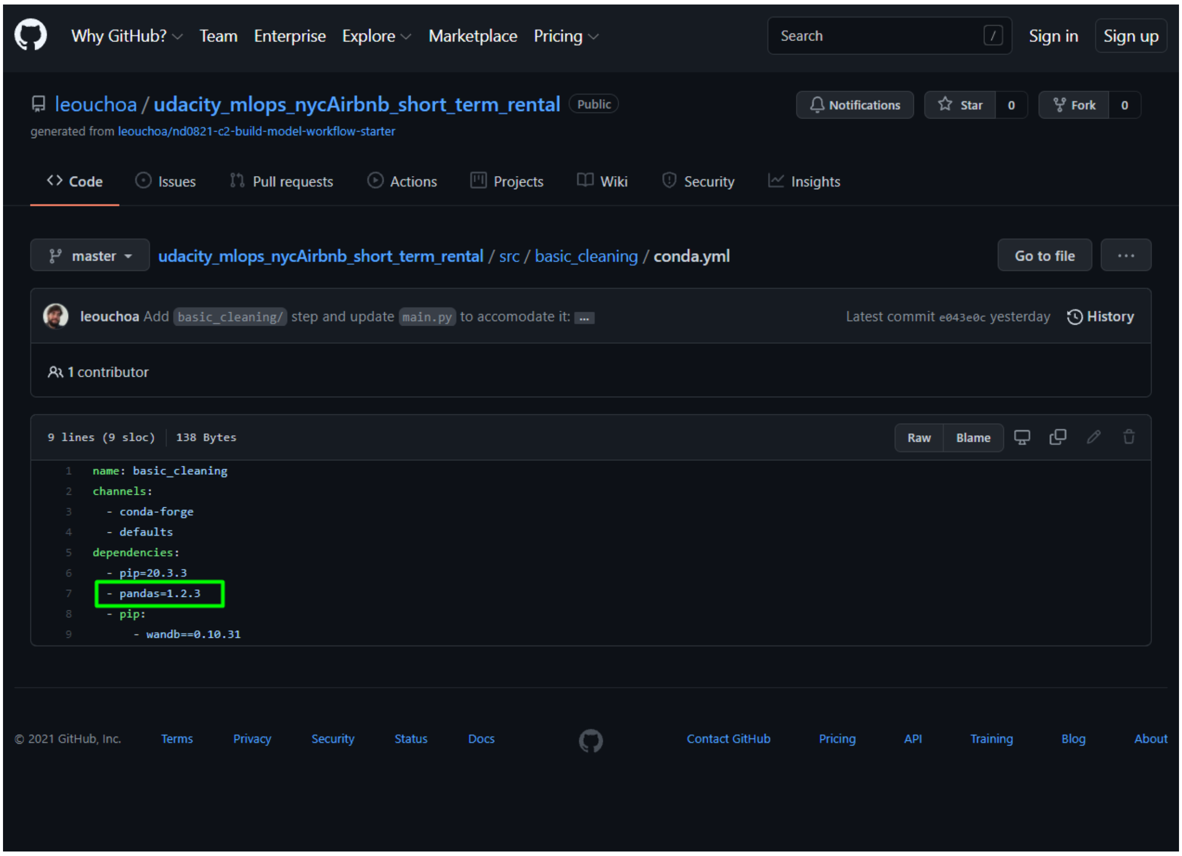


The conda.yml file has been updated to add the `pandas` dependency.

Data Cleaning

Pandas Dependency

You have set the pandas dependency in your *.yml* file!



Add docstrings and the proper type to all parameters, both in the script and in the MLproject file.

Data Cleaning

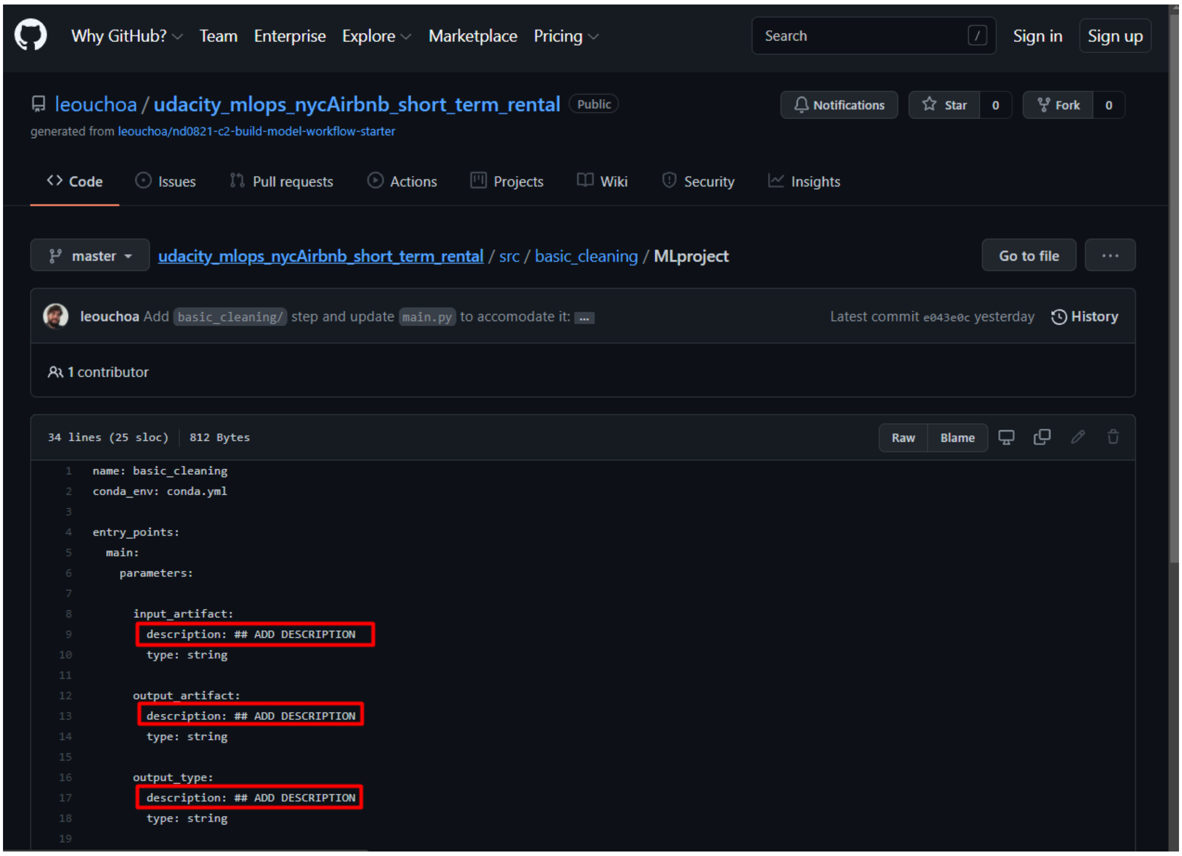
MLproject docstring

You are almost there! 😉

You should add the docstrings and proper type all parameters.

I would like to share some useful information to boost your learning path:

Create Reusable ML Modules with MLflow Projects & Docker



The basic_cleaning step re-implements in a MLFlow step the data cleaning you performed during the EDA. It should be added to the main.py file and run without errors.

In the main.py file all parameters are taken from the configuration file, and not hard-coded.

Data Cleaning

The *"basic_cleaning"* Step - MLFlow

Great job! 😉

In your *main.py* file, all parameters are coming from the configuration file and they are not hard coded. It is very important in the MLFlow framework so we can reproduce your code faster.

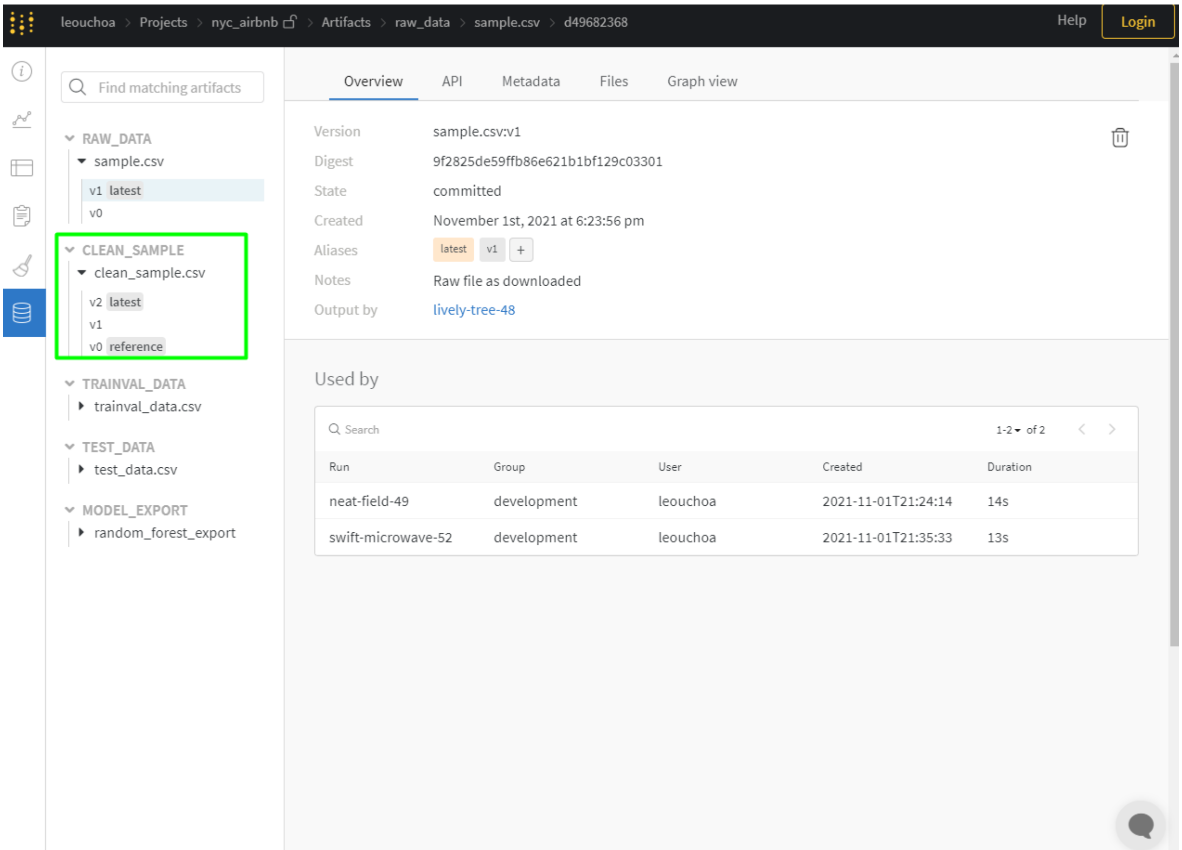
At the end of the run of this step, there should be a clean_data.csv artifact uploaded to W&B.

Data Cleaning

The *clean_data.csv* Artifact

Awesome! 😄

At the end of the run of this step, it is possible to check the *clean_data.csv* artifact in the W&B.



Data Testing

In W&B, manually add a tag called “reference” to the latest version of the clean_sample.csv artifact.

Data Testing

Tag *reference*

You have correctly added the tag *reference* to the latest version of the *clean_sample.csv* artifact.

Implements the test_row_count and the test_price_range tests in src/data_check/test_data.py.

The added tests are checking respectively for a proper size of the dataset, and for a proper price range.

Data Testing

The *test_row_count* Test

Perfect!

You have implemented the tests correctly!

The pipeline runs after this step, and all the tests pass.

Data Testing

Test Pepiline

Awesome! 😉

The pipeline runs after all tests and you have passed in all of them!

Data Splitting

Adds the train_val_test_split component to the main.py file.

The train_val_test_split has been provided to you. You can just add it to the `main.py` file and fill in the parameters appropriately.

Data Splitting

The *train_val_test_split* Component

You have added the *train_val_test_split* component to the main file.

The pipeline runs. At the end there should be 2 new artifacts on W&B: trainval_data.csv, test_data.csv.

Data Splitting

Two New Artifacts Added to W&B

Great job! 😄

At the end of the run the two new artifacts are on W&B.

Train the Random Forest

The `src/train_random_forest/run.py` script is completed.

When checking the script, there should be the following steps in the script, marked by clear comments:

1. Download the train data using W&B.
2. In the get_inference_pipeline function, implement a pipeline called `non_ordinal_categorical_preproc` with two steps: a `SimpleImputer(strategy="most_frequent")` and a `OneHotEncoder()` step
3. In the `get_inference_pipeline` function, create the inference pipeline called `sk_pipe` containing the preprocessing step and the Random Forest
4. In the go function, fit the pipeline.
5. In the go function, export the pipeline using MLFlow model export.
6. Upload the artifact to W&B
7. Log the variable MAE to W&B

Train the Random Forest

Script is Complete

Perfect! 😄

Your *run.py* at `src/train_random_forest/run.py` is complete and using the `sklearn Pipeline`.

The pipeline again runs successfully.

The `train_random_forest` step is added to the `main.py` file.

Train the Random Forest

Train Random Forest Step

In your `main.py` you have implemented the `train_random_forest` step.

There should be an artifact created on W&B called model_export.

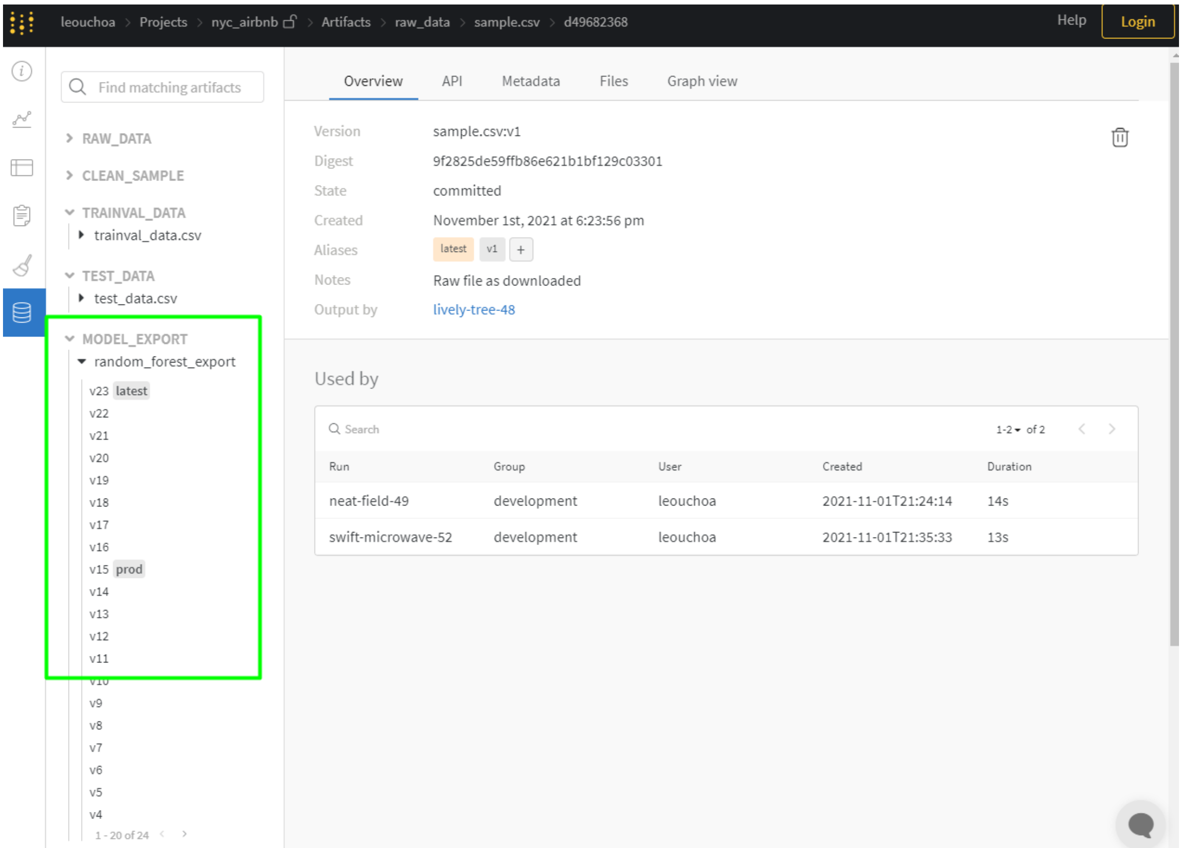
The model_export artifact should contain a MLflow sklearn serialized model.

Train the Random Forest

model_export Artifact

Great job! 😄

At the end of this step, there is an Artifact called *Model_Export* in your W&B!



Optimize Hyperparameters

Using the Hydra system, run a hyper-parameter search.

On W&B there should be the results of several (>2) training jobs with different hyperparameters.

Optimize Hyperparameters

Great job! 😄

Select the Best Model

Add the tag “prod” to the trained model with the best MAE.

Select the Best Model

Awesome job! 😄

You have added the tag `prod` to the trained model with the best MAE.

To boost your learning, there are two nice articles I would like to share:

- How to Choose Right Metric for Evaluating ML Model
- Evaluation metrics & Model Selection in Linear Regression

Test Set Verification

Implement the `test_regression_model` function in the `main.py` file. The test_regression_model is provided just as in the “data splitting” step.

Verify that the performance is comparable to what was obtained against the validation set (i.e. no overfitting occurred).

Test Set Verification

test_regression_model Function

The `test_regression_model` function in the `main.py` file is implemented.

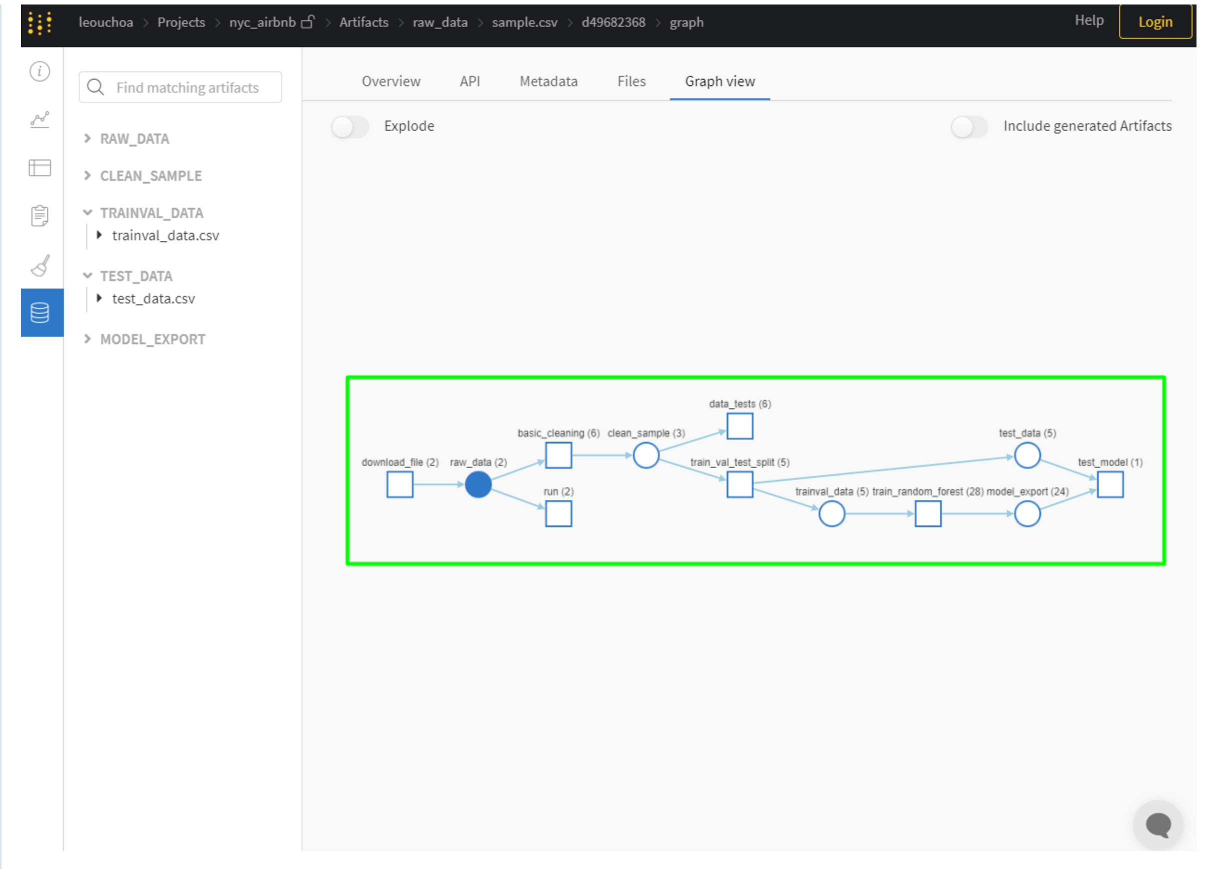
Visualize the Pipeline

Navigate to W&B, to the artifact section, then click on “Graph view”. The resulting visualization should show the pipeline properly organized. Refer to the reference plot in notes.md.

Visualize the Pipeline

Pipeline Graph View

Nice job! 😉



Release the Pipeline

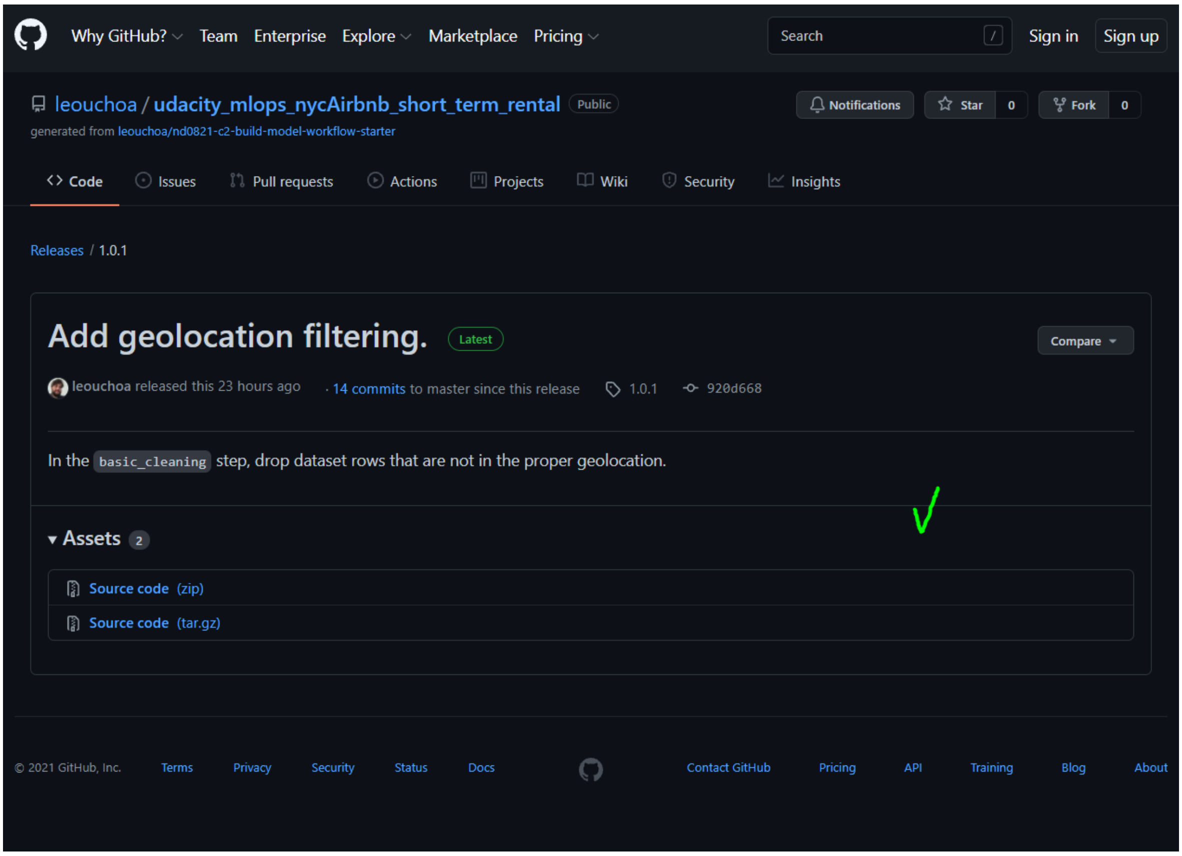
A release of the pipeline is cut from the Github repository, with version 1.0.0 or similar (if you need more trials, you might assign versions like 1.0.1 or 1.0.2, which is totally fine).

Release the Pipeline

Pipeline Release on Github

Perfect! 😄

You have released your pipeline in your Github account.



Train the Model on a New Data Sample

Run the released pipeline on a new sample of data, sample2.csv. The first version 1.0.0 (or similar) should fail, because there is a data problem in sample2.csv.

Train the Model on a New Data Sample

Released Pipeline Run

You have correctly set your project. 😄

Implement a new cleaning step that removes data points that are outside of the area of NYC in `basic_cleaning`.

Train the Model on a New Data Sample

New Cleaning Step

Perfect! A new step cleaning is implemented.

After adding the new cleaning step and committing and pushing to the repository, release a new version (for example, 1.0.1).

Train the Model on a New Data Sample

New Release

Great job adding the new cleaning step. You have commited and pushed to the repository. You have released a new version as requested. 😄

Re-running with the new release should produce a new trained model.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START