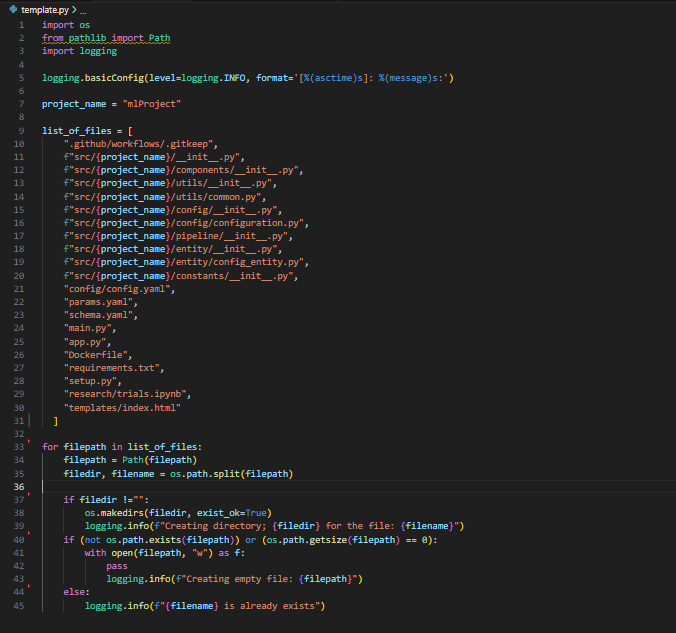
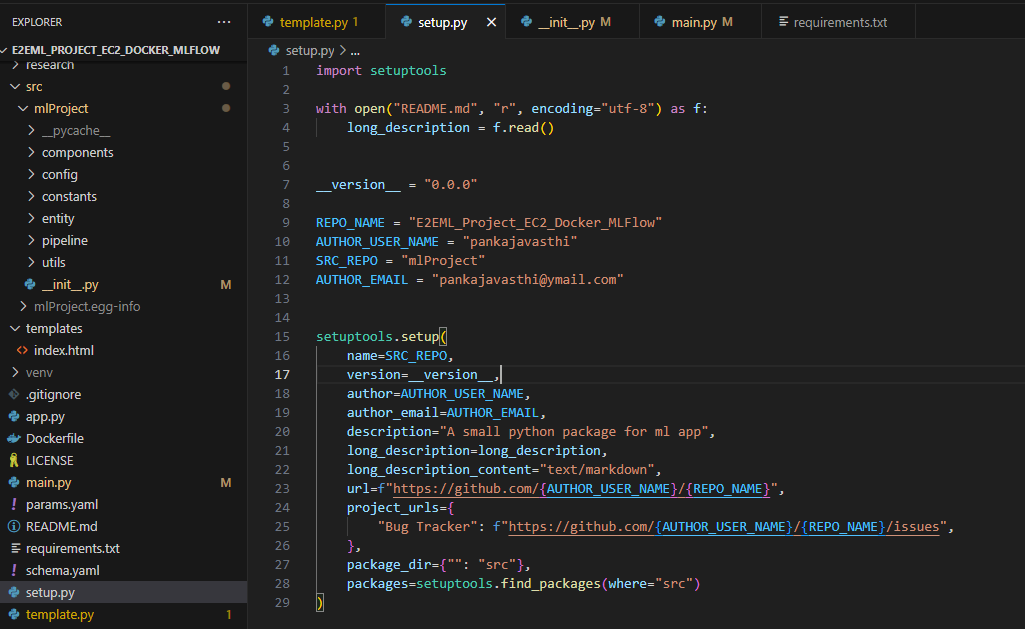
**E2E Machine Learning Project using GIThub Actions(CI/CD), Flask for UI, Docker to Containerize, Amazon EC2 for Deployment in Cloud with ECR for Private Container Registry to move our Docker File MLFLow tool using Dagshub for Model evaluation & Experiments**

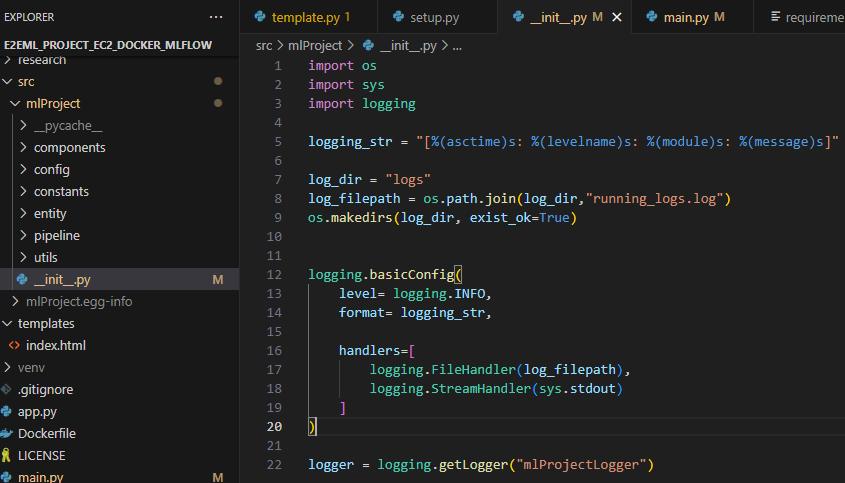
**Template.py file to create folders and files for our project**



5)**Setup.py** - Before running requirement.txt, run setup.py to make all source folder as local packages using \_\_init\_\_.py constructor



6) **Logger file** – Create customer logger with below mentioned code to generate log in log folder and at terminal.



7) **Util Folder-** Util folder is create to create most common function which will be required frequently in our project. Those functions, which are required frequently in our components like reading yaml file we can create in common.py file and call it from here. Always require in Modular Coding style. Here we have functions a) Reading Yaml File b) Creating Directories like artificats, dataingestion, datavalidation, modelcreation c) save\_json, load\_json, save\_bins, load\_bins & get\_size. Kinldy refer code as we are using decorators like @ensure\_annotations and other functions like box.Exceptions and ConfigBox functionality in detail.

**8) WorkFlow:** Below is the process and usage of various file and folders to initiate the workflow by following modular level and object-oriented programming concepts. Please see below details

1. Update config.yaml # Used to make all configuration oof our file

2. Update schema.yaml # We use Schema to define our variables used in data structure

3. Update params.yaml # All the parameter used in our project, will define here

4. Update the entity #

5. Update the configuration manager in src config

6. Update the components # Data ingestion, Validation,

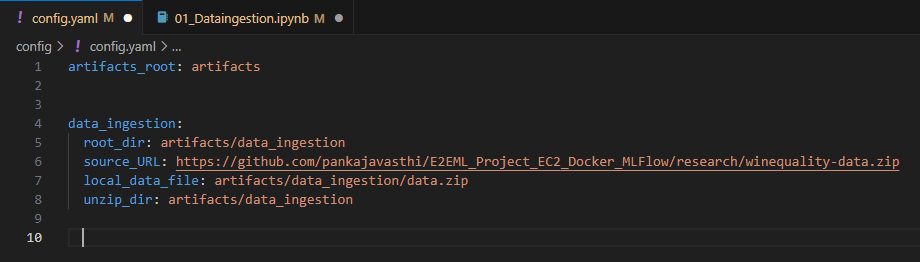
7. Update the pipeline # All components will be integrated through pipeline

8. Update the main.py # to maintain all code and run them centrally

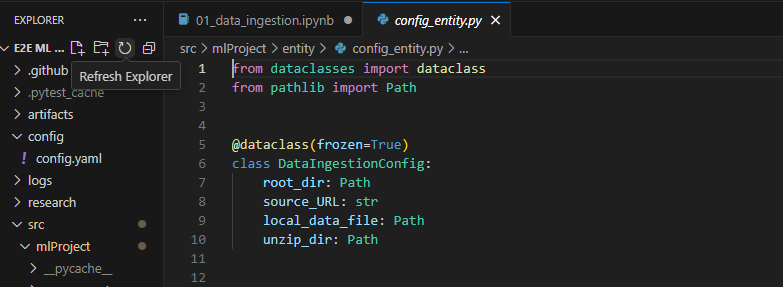
9. Update the **app.py # To create a flask app UI for our project**

**Now Let’s Follow above steps to understand more using Data Ingestion Process**

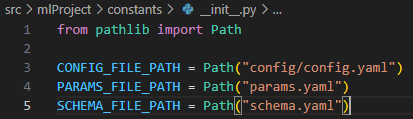
1. Create **config.yaml** file to create **artifacts** folder to keeps all output of all processes involved in ML project like Data Ingestion, Data transformation, Data Validation, Models training, Model Evaluation. Firstly, we mentioned only for Data ingestion.



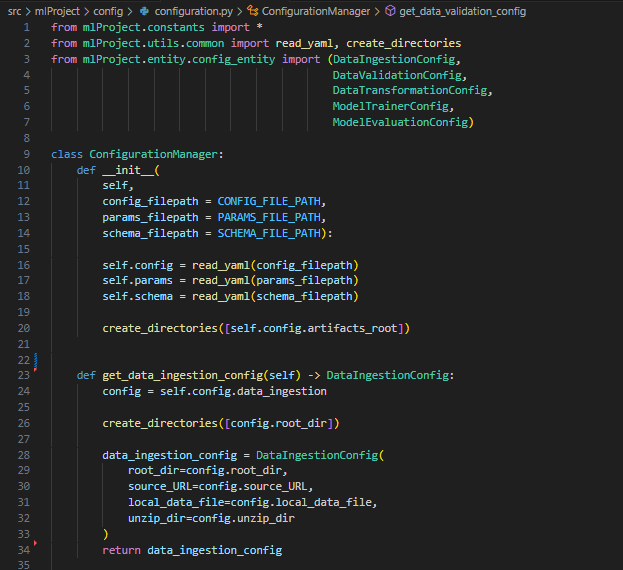
1. Step 2 **schema.yaml** will be updated in **Data Validation process** to match the variables of dataset.
2. Step 3 **params,yaml** file will be update with **model parameter** at the time of model training.
3. Step 4 inside **src/mlproject/entity** we have **config\_entity.py** file, used to set return type of function here is **@dataclass decorator** is used to get return type.



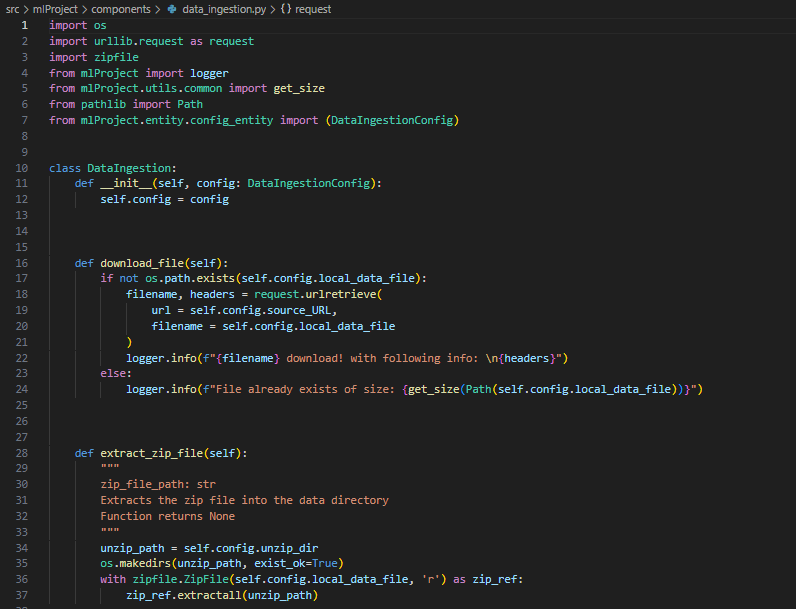
1. Step 5 inside **src/mlproject/constant,** in constructor file init.py, create path of all yaml files to read.



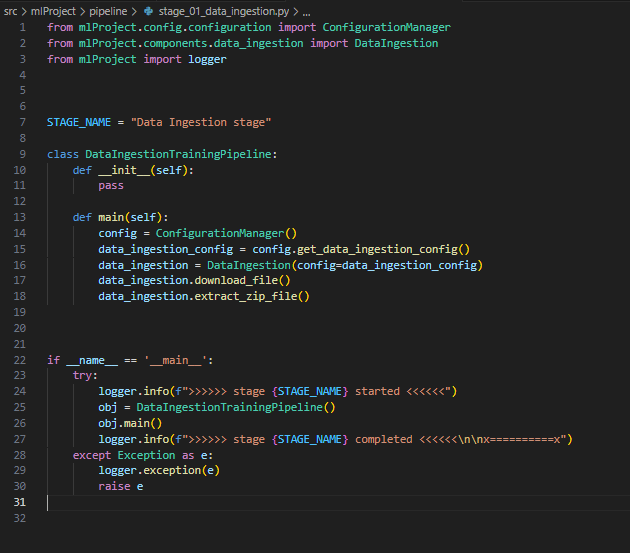
1. Now in **src/mlproject/config**, create **configuration.py** file to create configuration manager



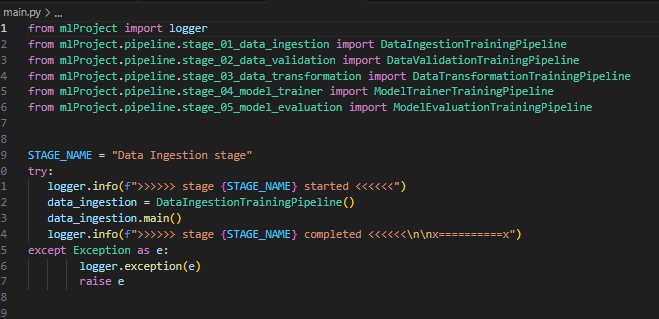
1. **Now create components for Data ingestion using configuration manager and common function**



1. Create **pipeline** from **src/mlproject** inside it **config/configuration.py** and **components/data\_ingestion.py, file name is stage\_01\_dataingestion.py**

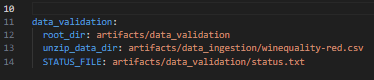


1. Now move to main.py and run from terminal after inserting these codes.

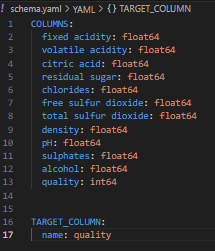


**Stage – 02 (Data Validation Stage)**

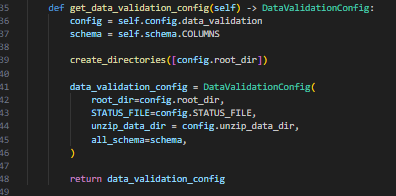
1. **config/config.yaml file update -**



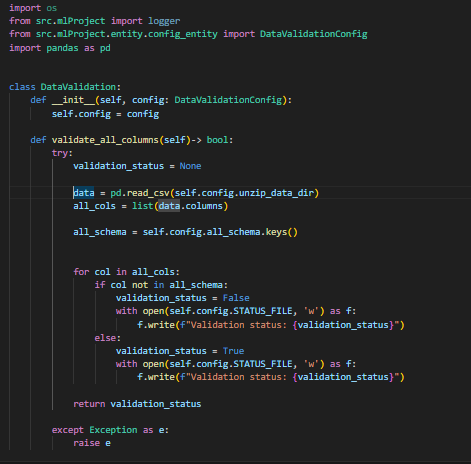
1. **Schema.yaml file updation -**



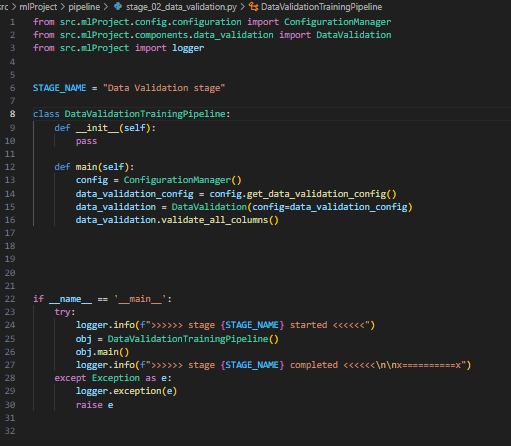
1. **src/mlproject/config/Configuration.py - file updation**



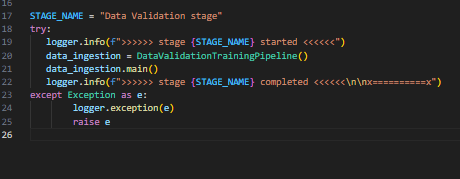
1. **src/components/data\_validation.py – file creation**



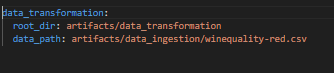
1. **src/mlProject/pipeline/stage\_02\_data\_validation.py – File creation**

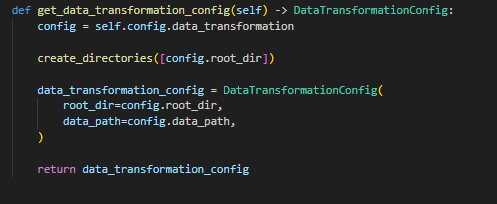


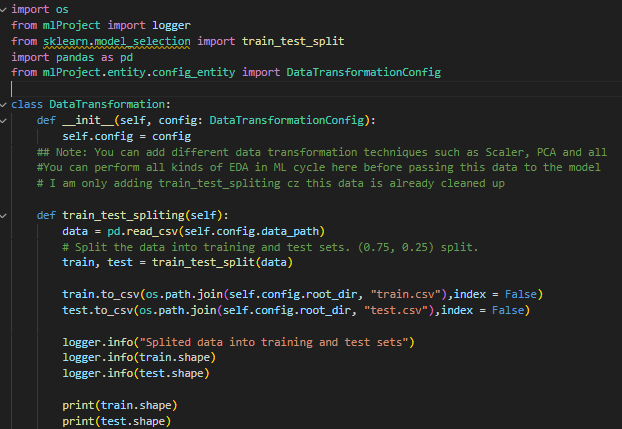
1. **main.py – File updation**



**Stage 03 - Data transformation**









**Stage 04 (Model Trainer)**

