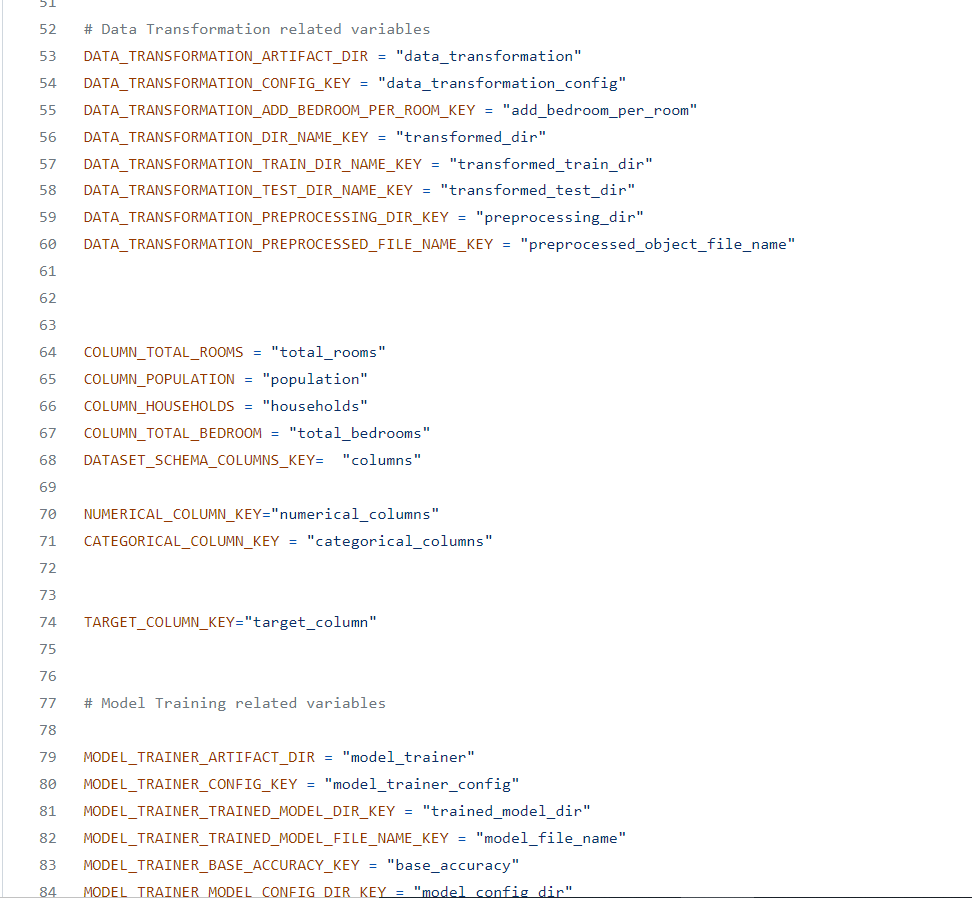
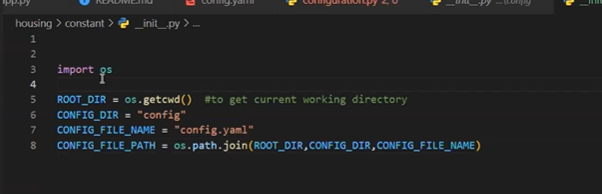
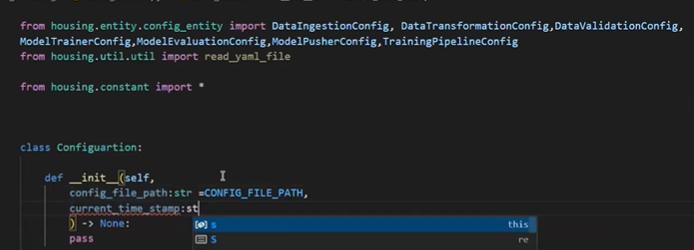
file



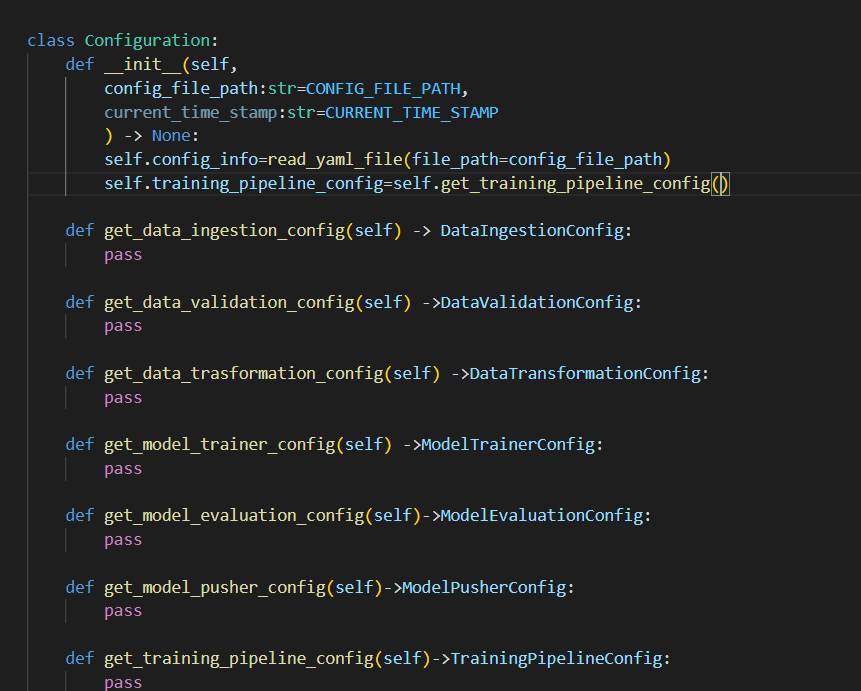






Now working each function in configuration class

First \_\_init\_\_



Config\_file path and current time stamp: getting from constants

Reading self. config\_info== from read yaml func in utils

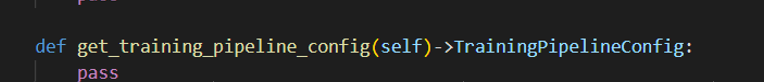
And

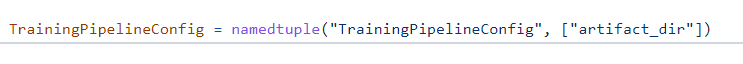
Self. Training\_pipeline\_config=self.get\_training\_pipeline\_config

Now we will work on

Self.get\_traininf\_pipeline\_config:

Last function in configuration



In entity I have written   


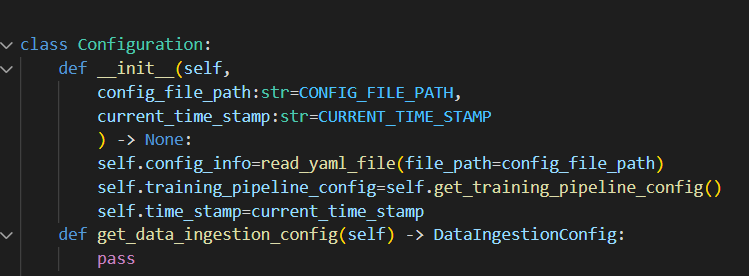
In trainingpipelineconfig- I have written artifact directory as artifact\_dir

When I write code

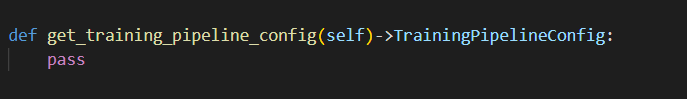


When I type. A

Automatically that name will appear I don’t need to remember those names

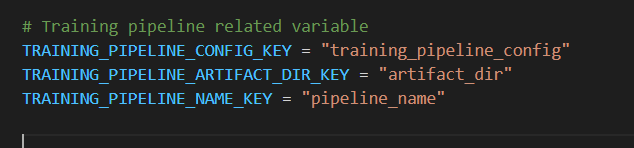


Now afterthis we will work on training\_pipeline\_config



Now create constants

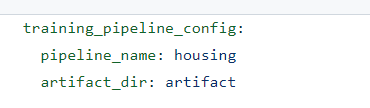
Related to training\_pipeline



Why we have given these names

In config.yaml of config file

The names are like



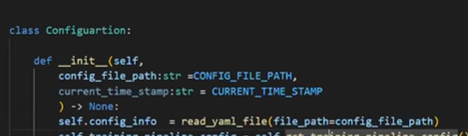
To get the exact values of keys mentioned in yaml file we have used key values as a values to constant variables

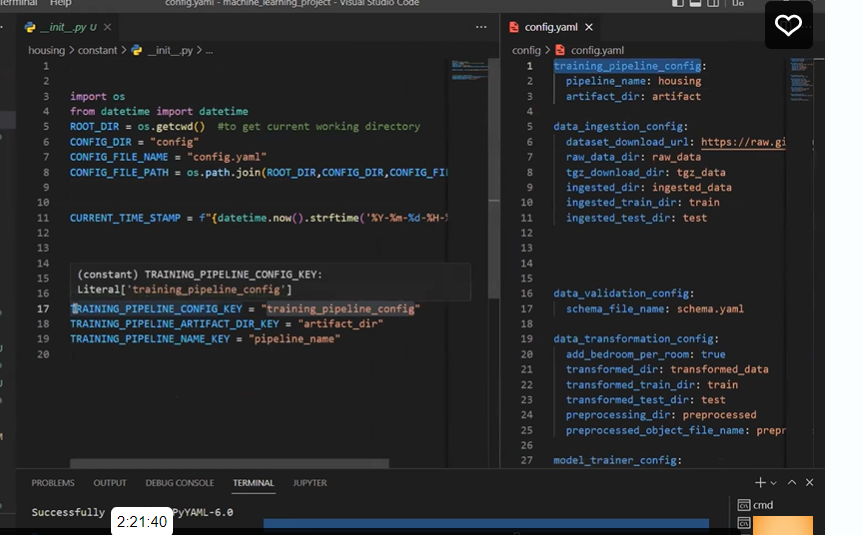
Here Constant variable is

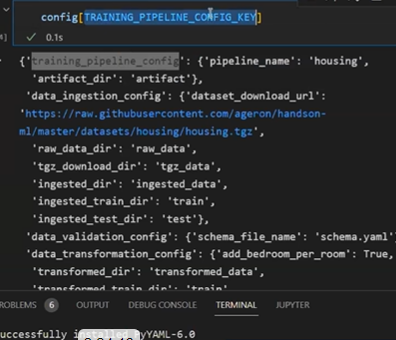
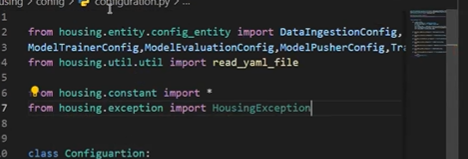


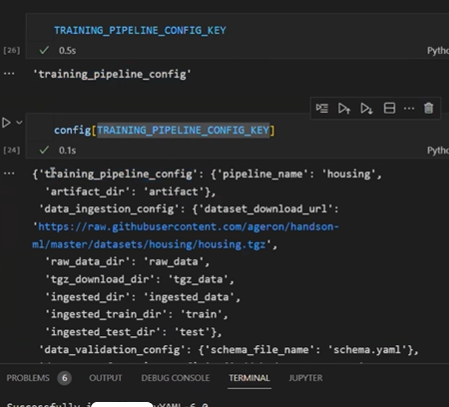
Name assigned to it is a key mentioned in yaml file 

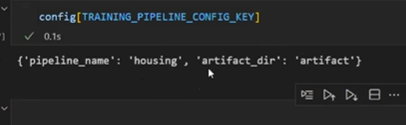
Which reads name house

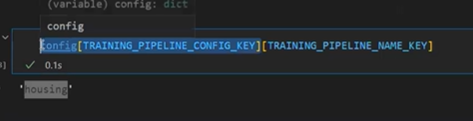


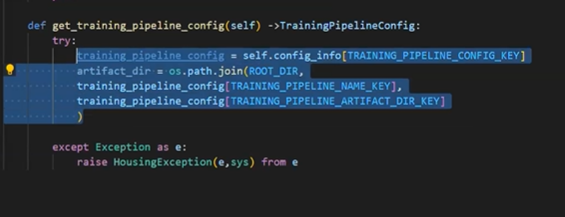






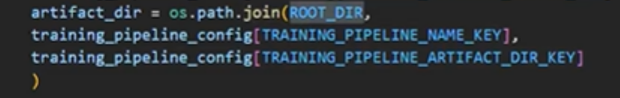






-- housing folder

This we get Housing

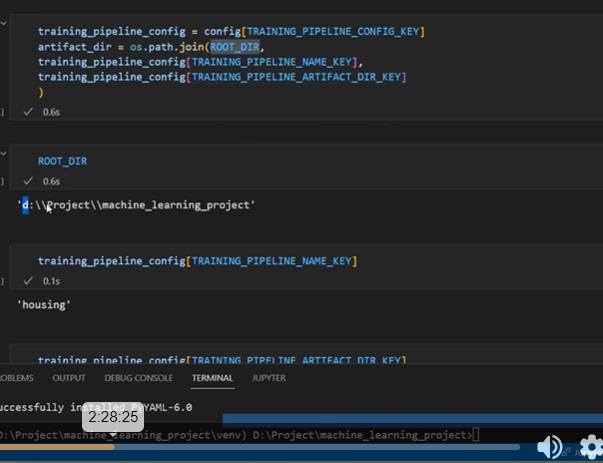


This we get

d:\\Project\\ machine\_learning\\housing\\artifact



artifact

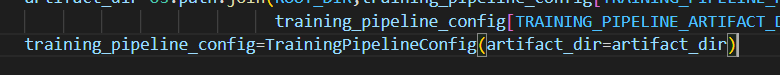


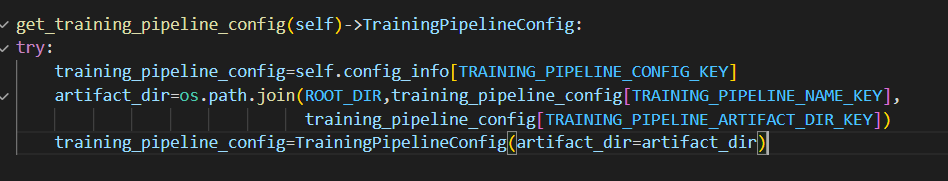
Configuration is just giving folder information

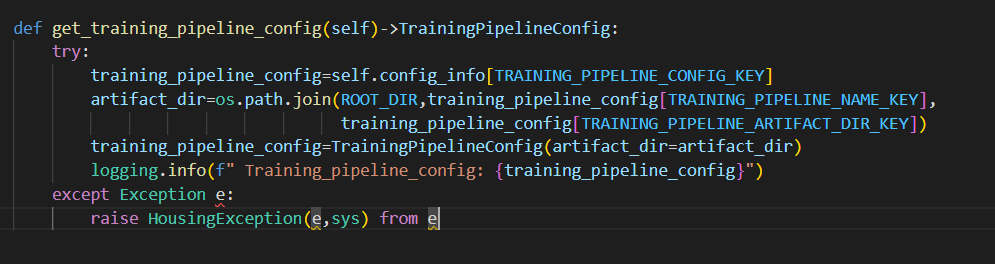
Configuration : nothing to do with creating or updating folder /updating

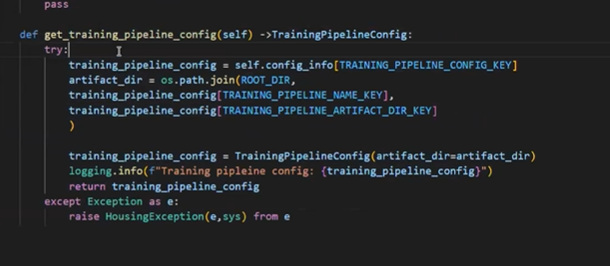
We are just giving path / info of folder/ what info we need to give

Now we are creating namedtuple

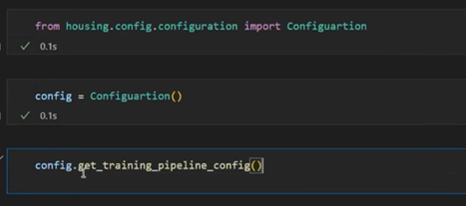








Checking in jupyter notebook



When we call like that we get

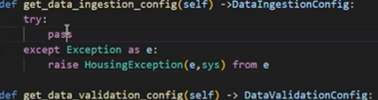
TrainingPipelineConfig(artifact\_dir=’d:\\project\\machinelearning\\housing\\artifact’)

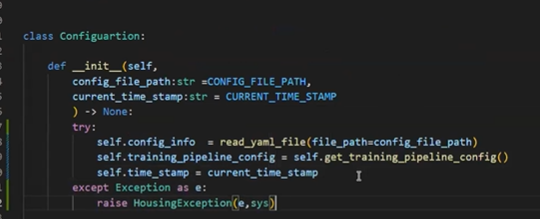
Flow:

Configuration.py using yaml file then referring entity and then configuration

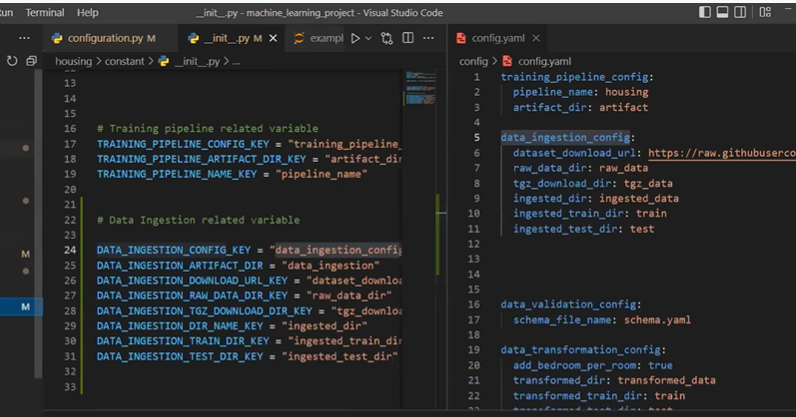
Now data\_ingection\_configuration:

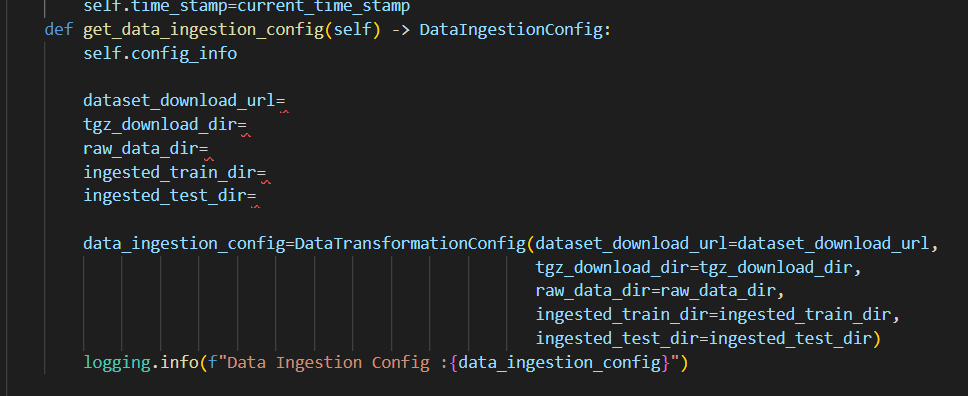


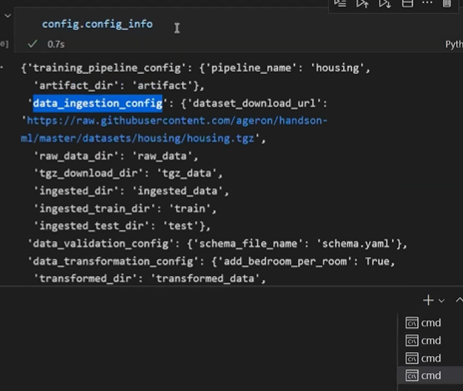


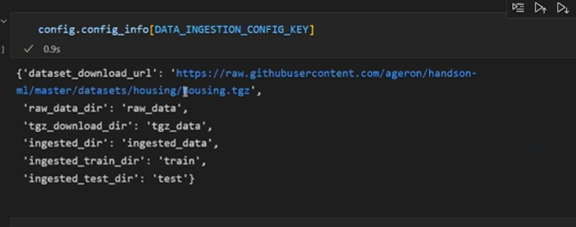


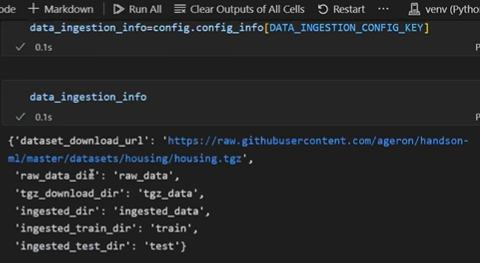
Define constants- data ingestion



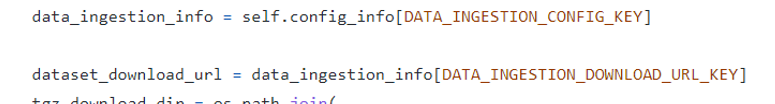






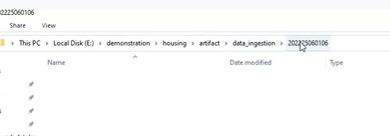




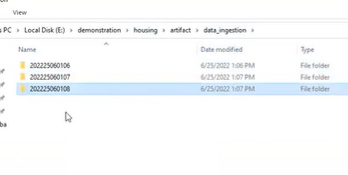




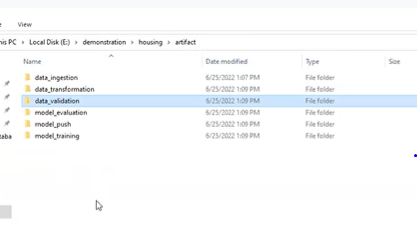


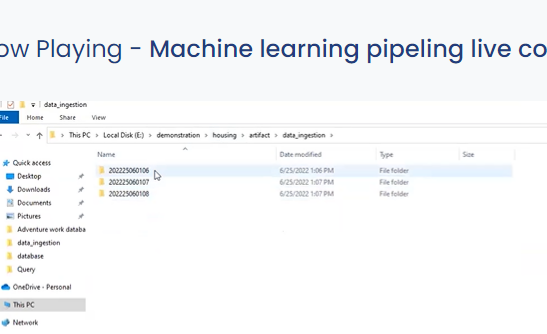


Folder will be created based on time stamp

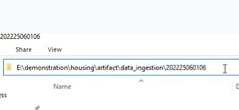
----

Whenever we run it creates those many folders





For data injection artifact path should be



First we need artifact directory



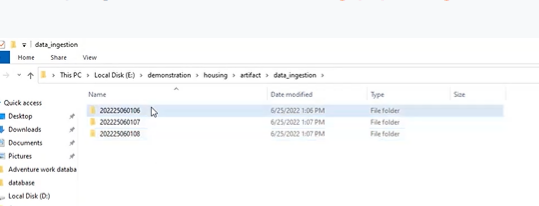
\

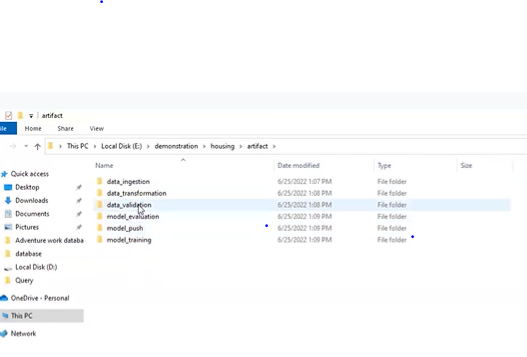
Data ingestion\_config

In artifact folder we have multiple folders for each components

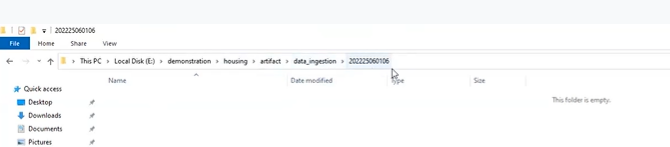
At each timestamp

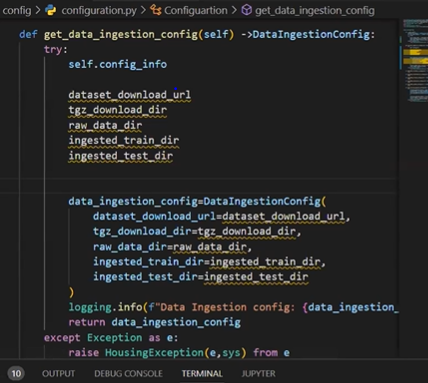


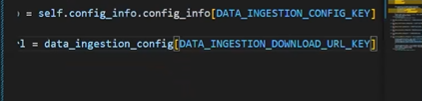


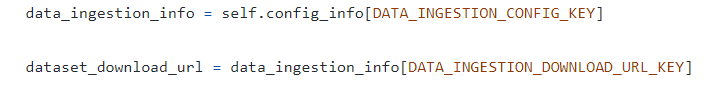


Dataingestion artifact dir





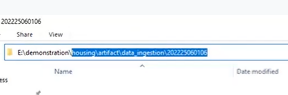


within artifact only we need to create all other component artifacts with timestamps

First I need artifact directory

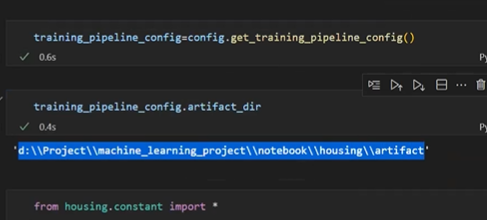
Housing-artifact

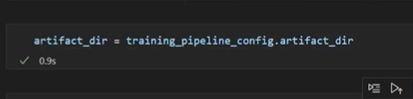
Where as for dataingestion -the artifactdirectory is-



Till training pipeline

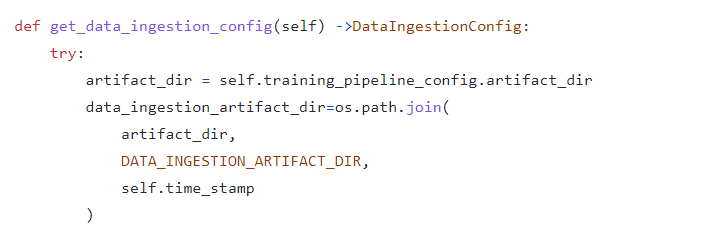
We get artifact

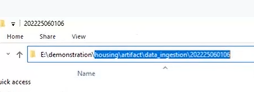




Now we need to add data ingestion and timestamp

From constant file we can get dataingestion and timestamp

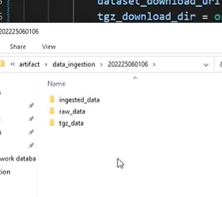




We get this path

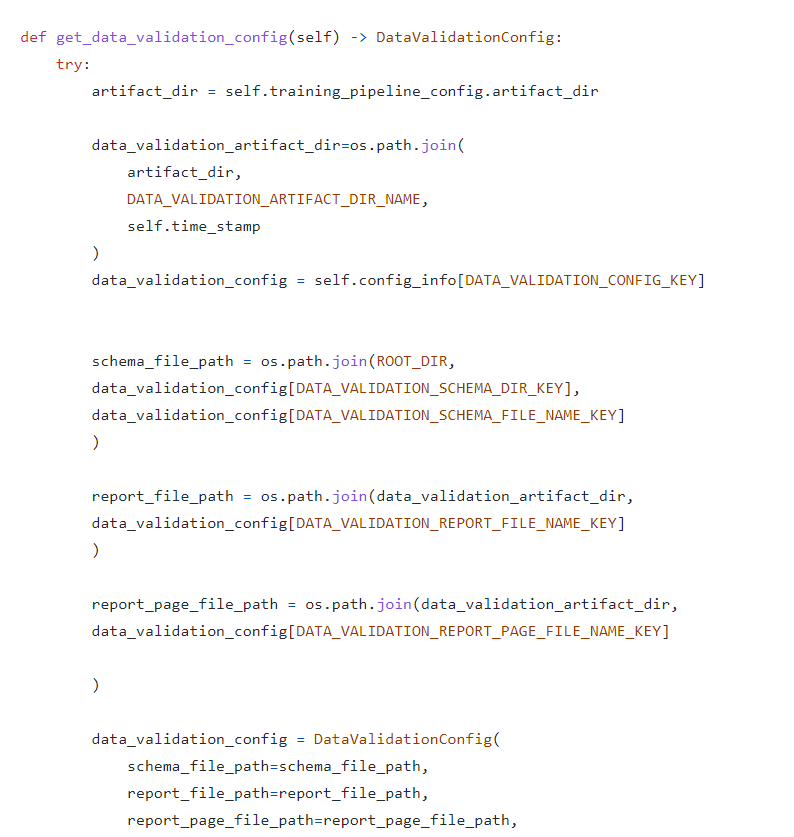
Now raw dir

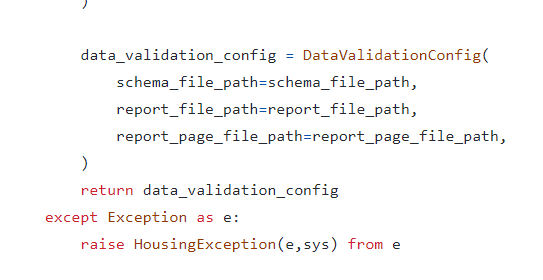




Train and test with in ingested data

Now: validation











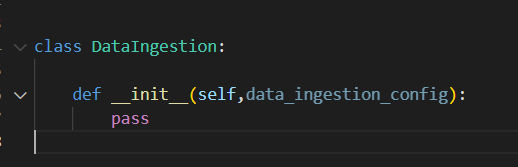




Now we will work with under component

Dataingestion.py

Create class



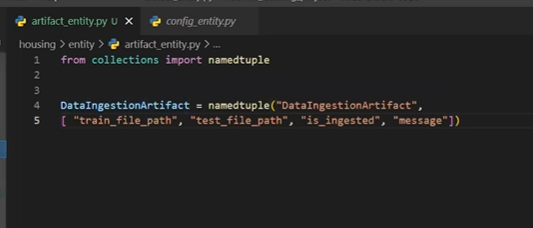
Class 61:

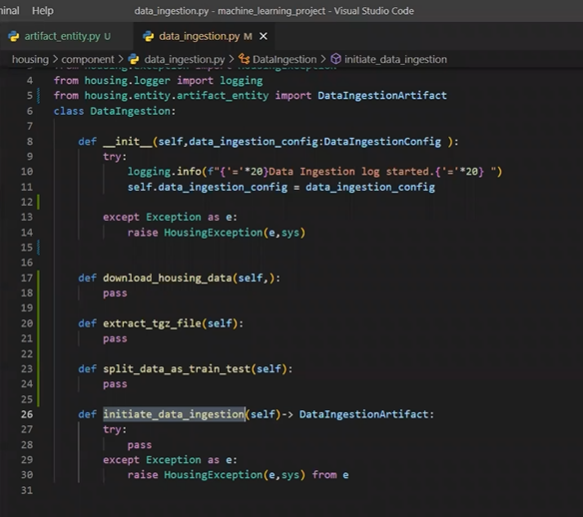
Artifact\_entity.py

Create an another file in entity folder

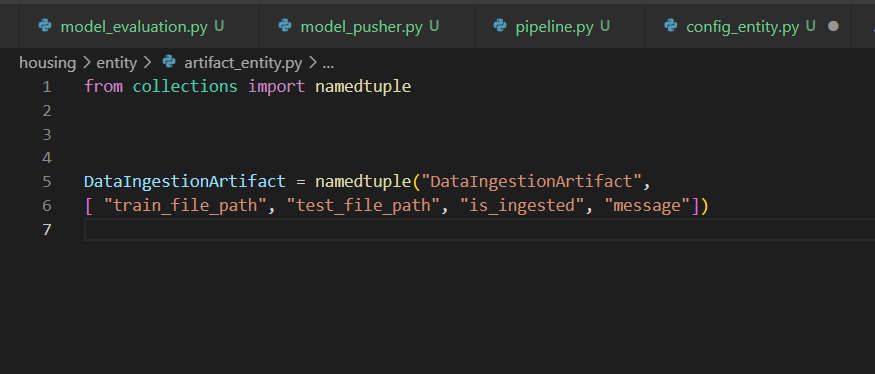
Datainestionartefact:

To save save 1. Trainfile, test file location(datavalidation and transformation, is\_inserted to know successful, messgage , )





Now create artifact\_entity.py in Entity folder



Is\_injested: to know whether it was successful or not