**Machine Learning Project SOP**

**Machine Learning Project Life Cycle:**

Github:

* Create a repository in github.com
* Repository name:
* Description:
* Public/Private
* Add a README file:
* Add .gitignore
* Choose a license

Once created the repo just clone the repo to your local system

command prompt:

* open cmd
* go to the path where you want to create a clone repo. [cd D:\Bala\Full Stack data science ]
* execute the below cmnd:
* git clone https://github.com/BalaM14/Machine\_Learning\_Project.git

Launch VScode:

* go to the repo location. [cd D:\Bala\Full Stack data science\ Machine\_Learning\_Project ]
* execute the below cmnd:
* code .

Step1: Create a conda environment # conda create -p venv python==3.11 -y

Step2: Activate the created virtual environment # conda activate venv/

Step3: Create the requirements.txt file

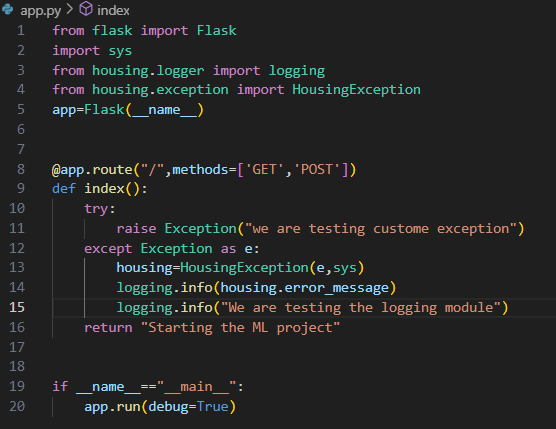
Step4: execute the install command to install the required libraries # pip install -r requirements.txt

Step5: Check the environment folder name is in git.ignore file. So that while pushing to github the env folder will be ignored

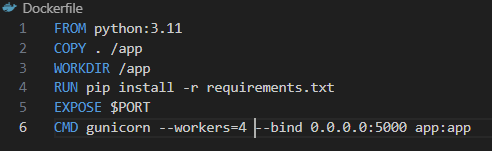
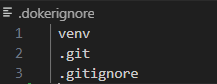
Step6: upload all the updates & changes into the git repo by using git commands.

* # git add <filename> -> To add single file
* # git add . -> To add all the files
* # git status -> To view the status of the files
* # git log -> To view the logs of the git cmnd executed
* # git commit -m "message" -> To commit all the changes to the repo
* # git push origin main -> To push the changes into the git repo & make them available in github
* # git remote -v -> To check the remote Url pointing for push/fetch commands
* # git branch -> To view the branch name
* # git diff -> To see the difference b/w the previous commit and new changes

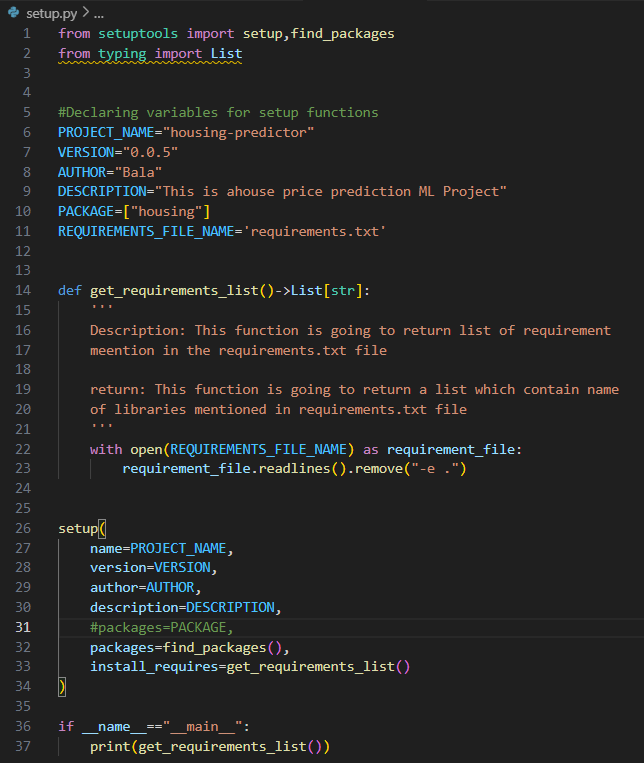
Step7: Create app.py sample file to test the application deployment using Flask.



Step8: Create a Dockerfile & .dockerignore file to create a docker image with all the required commands.

Step9: Create a setup.py file to invoke the requirements.txt file to install all the required packages.



Step10: Execute the setup.py file by using the command # python setup.py install

requirements.txt -> At the eof we are using the "-e ." for installing the python packages as well from all the modules created by user.

* without "-e ." -> pip install -r requirements.txt

Successfully installed Flask-3.0.2 Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.1 blinker-1.7.0 click-8.1.7 gunicorn-21.2.0 itsdangerous-2.1.2 joblib-1.3.2 numpy-1.26.4 pandas-2.2.1 pytz-2024.1 scikit-learn-1.4.1.post1 scipy-1.12.0 threadpoolctl-3.3.0 tzdata-2024.1

* with "-e ." -> pip install -r requirements.txt

Requirement already satisfied: six>=1.5 in c:\users\balaj\appdata\roaming\python\python311\site-packages (from python-dateutil>=2.8.2->pandas->-r requirements.txt (line 5)) (1.16.0)

Installing collected packages: housing-predictor

Running setup.py develop for housing-predictor

Successfully installed housing-predictor-0.0.5

* findpackages() -> equivalent to "\_e ." where both are used to install all the user created module packages

* if we want to install only the user created module packages then we can use the command :

# pip install -e .

* if we want to install the external packages as well as user created module packages add the "-e ." at the eod of requirements.txt -> # pip install -r requirements.txt

Step11: Create a below folder structure under the main folder having the \_\_init\_\_.py in each and every folder.

-> exception

-> logger

-> component

-> config

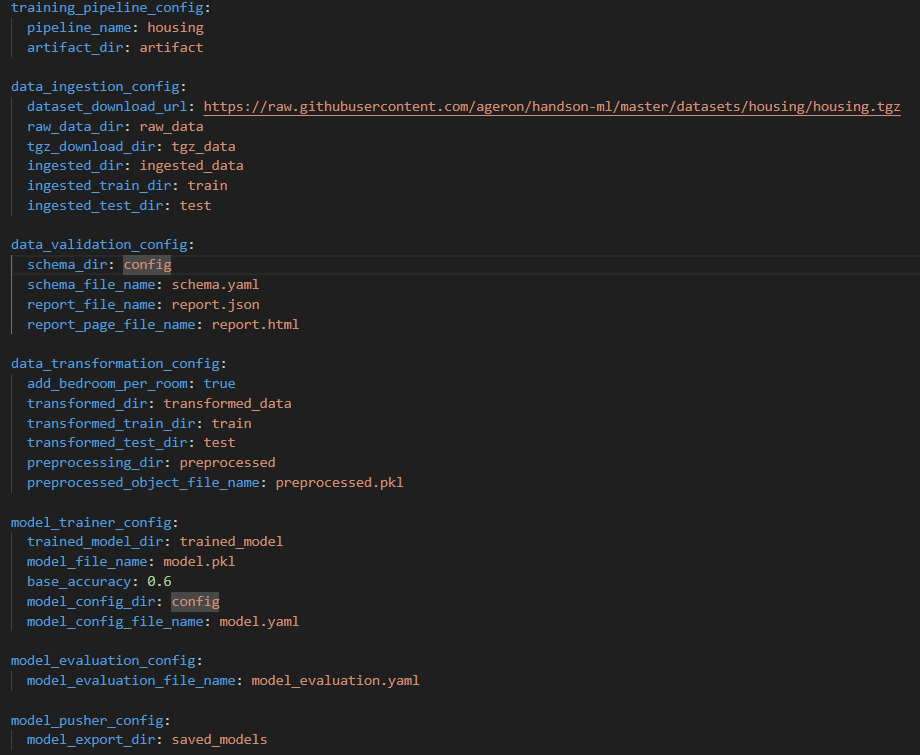
-> pipeline

-> entity

-> constant

-> util

Step 12: Create a notebook & config folder in main path. In the notebook folder create a sample.ipynb file for testing and workthrough. And in config folder create a “config.yaml” file

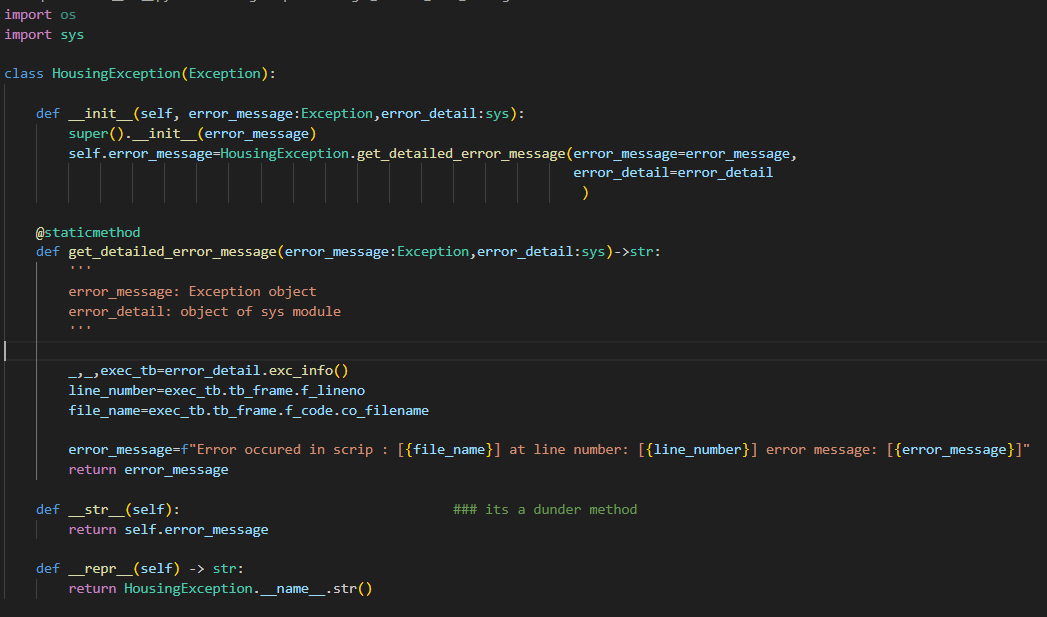


Step13: Start coding for logger & exception part.

* Logger:

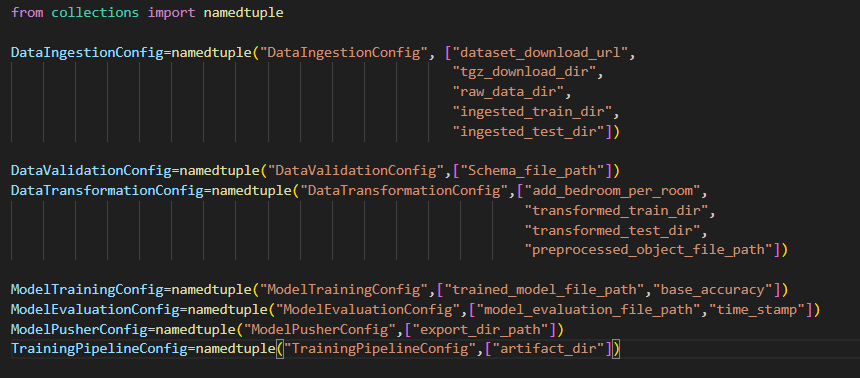


* Exception:

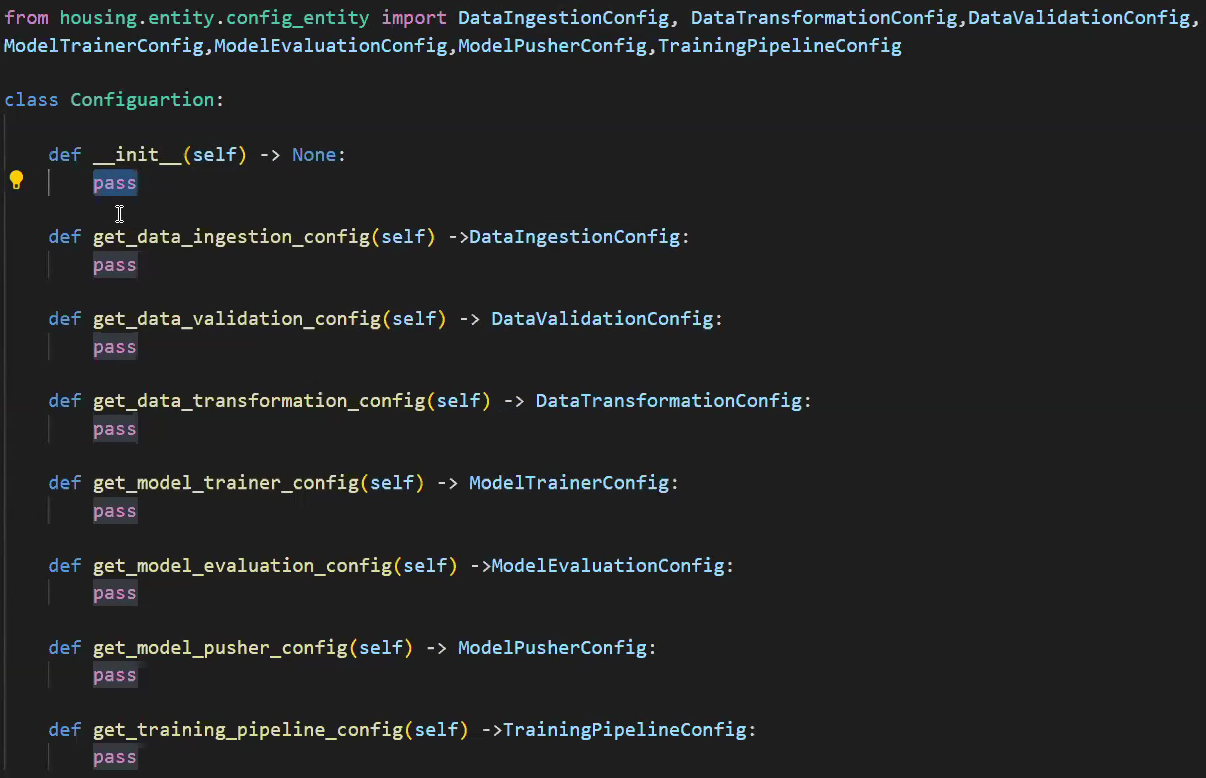


Step14: Start working on entity and config folder

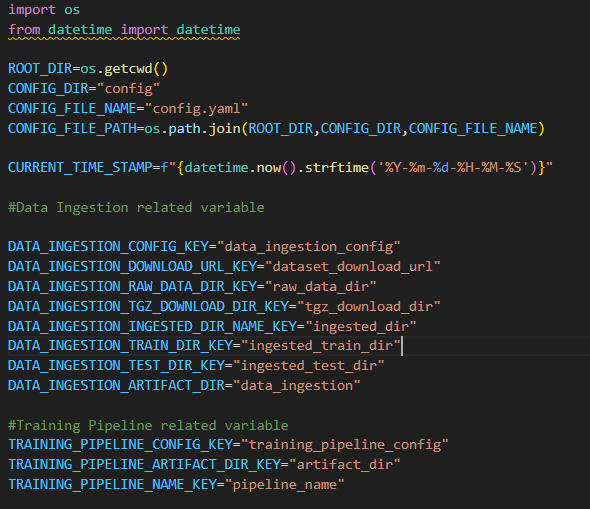
Create the “config\_entity.py” file under the housing/entity/



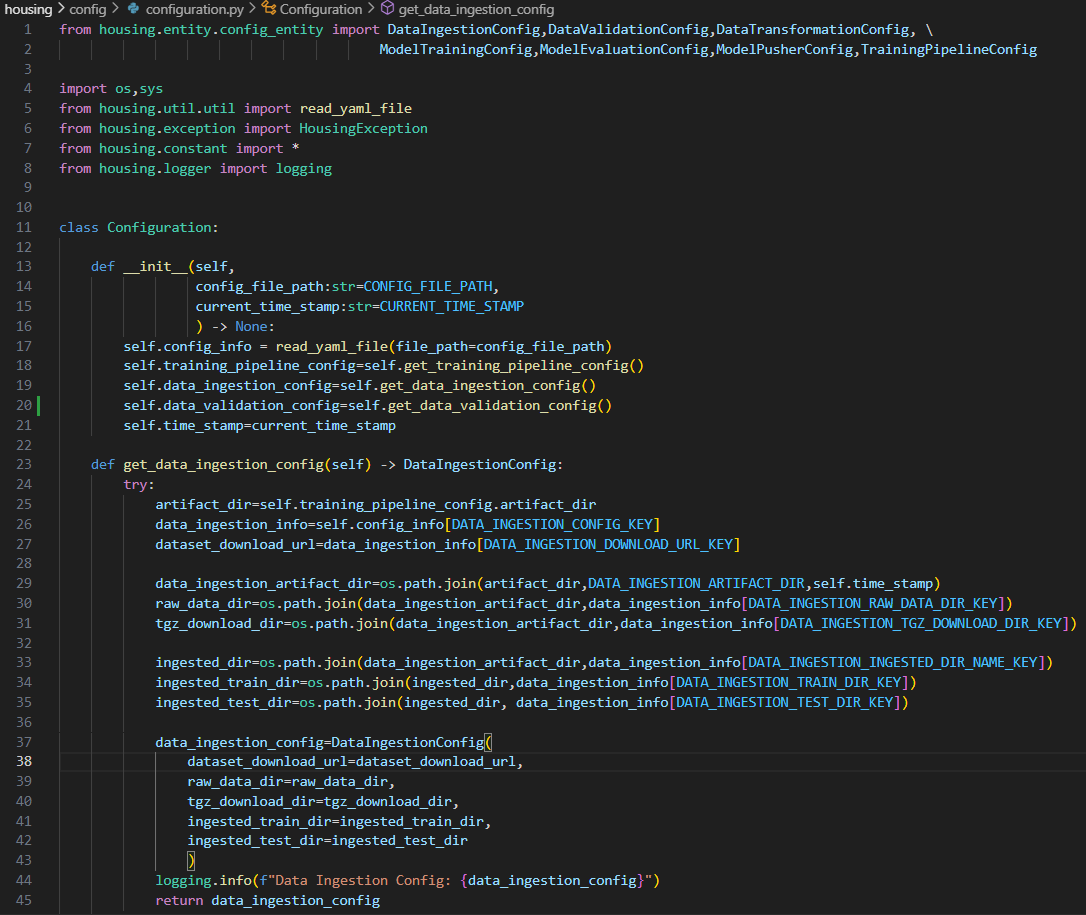
Create the “configuration.py” file under the housing/config/



Step15: Create the \_\_init\_\_.py file under constant folder with all the user defined constant variables

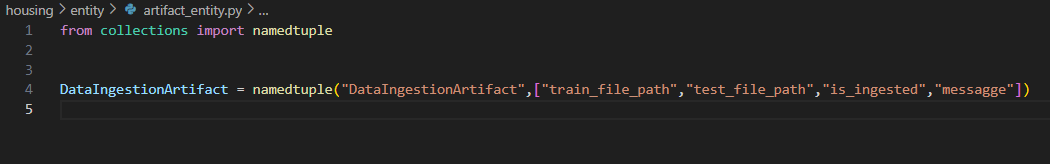


Step16: Update the “configuration.py” with the constant variables. Same way we need to do for all the methods.

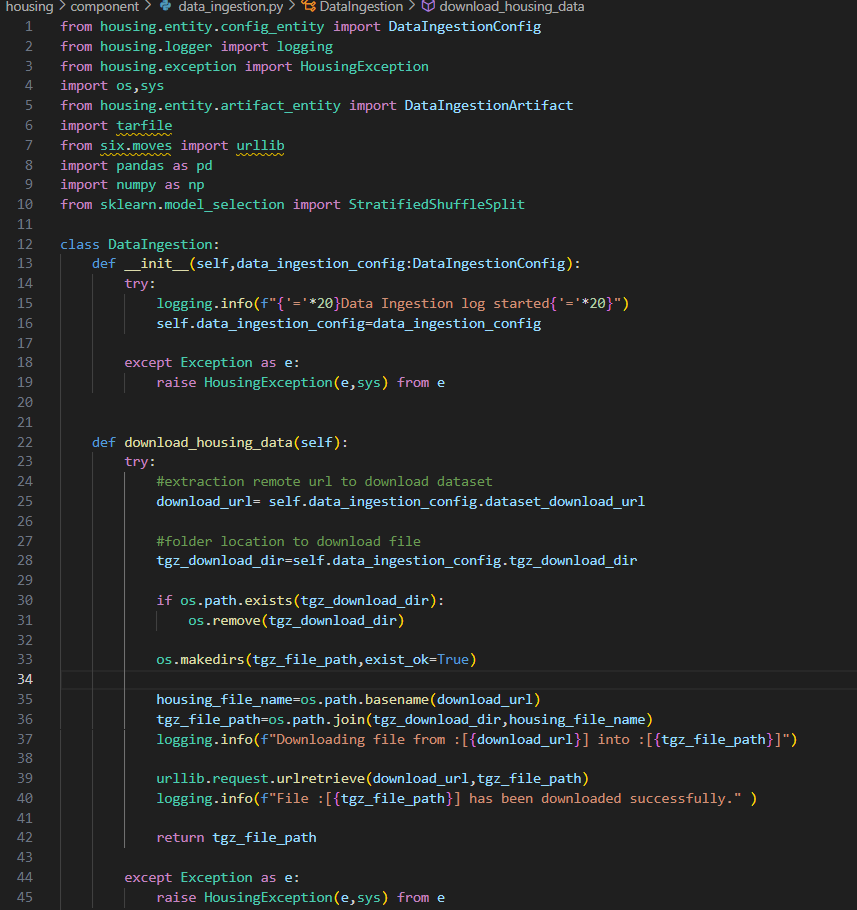


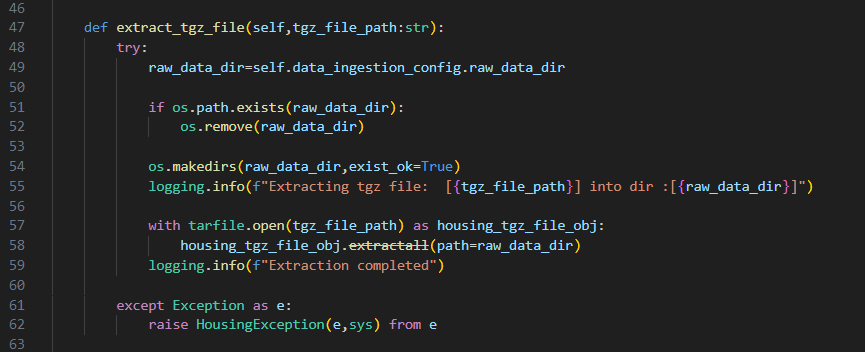
Step 17: Create a file “artifact\_entity.py” under the “entity” folder.

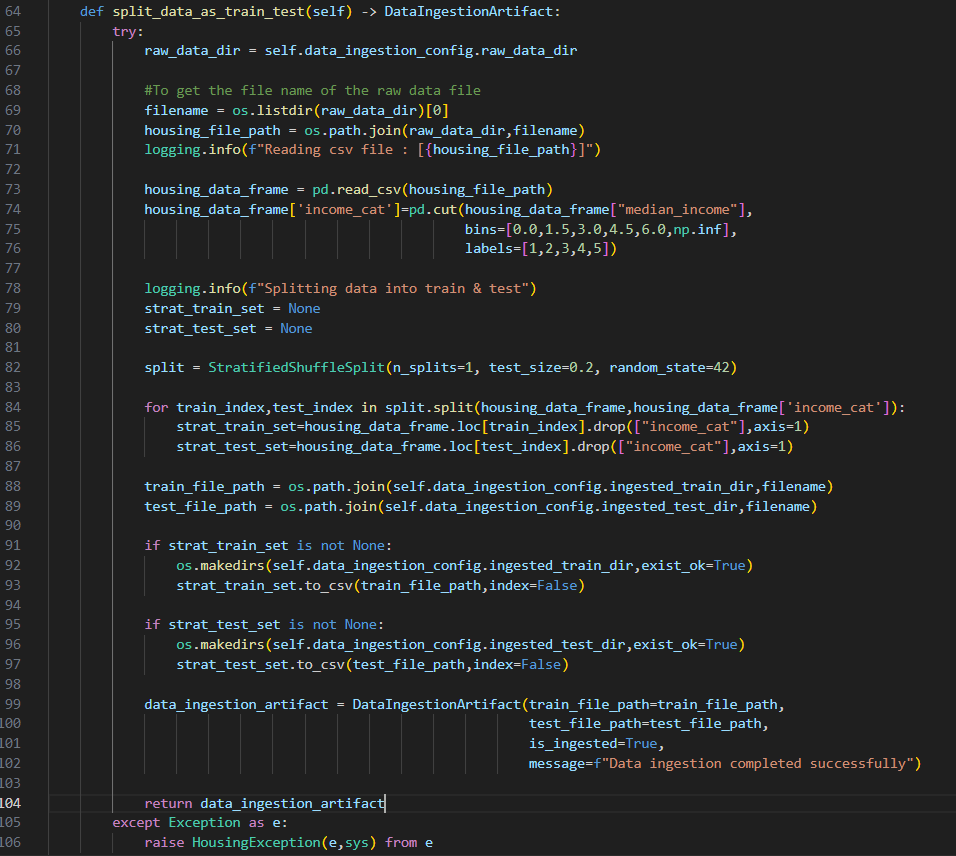
* config\_enity => used as an input config for all the components
* artifact\_entity => used as an ouput config for all the components

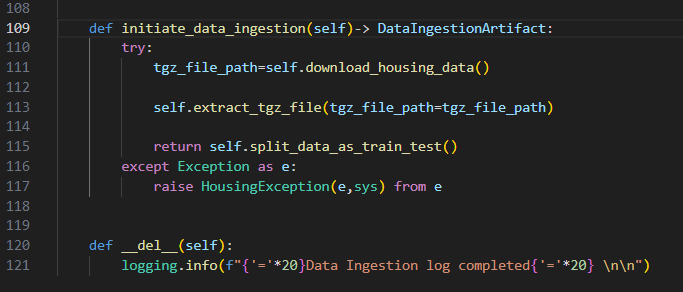


Step 18: Start the coding on “data\_ingestion.py” file









Step 19: Start coding on “pipeline.py”

