**ML Project**

1. We first opened an Anaconda prompt
2. In the anaconda prompt we typed cd with the project folder location
3. Then we typed code . to launch VS code instance
4. To ensure that I sync with my GitHub repository I need to first open a New Terminal
5. To create a virtual environment, I tried this code: conda create -p venv1 python==3.8 -y and then ran conda activate venv1
6. Now I created a Readme.md file in the project folder and edited that with ## End to end Machine Learning project line and saved it
7. Then I used: git add README.md toad to git
8. Then I did git commit -m "first commit" to commit it to the git
9. To create a branch and sync with git I used the code:

git branch -M main

git remote add origin https://github.com/PretomGhosh/MLProject.git

1. Then I did git push -u origin main to push it to the git
2. Then I created a .gitignore and selected python for the .gitignore
3. I created a setup.py to create my project as a package
4. I created a requirements.txt file

In setup.py I wrote the following codes: from setuptools import find\_packages,setup

setup(

name = 'MLProject',

version = '0.0.1',

author = 'Pretom Ghosh',

author\_email = 'pretomghoshcuet@gmail.com',

packages = find\_packages(),

install\_requires = ['pandas','numpy','seaborn']

)

1. Setup.py needs to look \_\_init\_\_.py files to search and install it as a package, we have created src folder and created a \_\_init\_\_.py file in it. We will basically build the project using the src folder and what’s inside that folder
2. We will now create a function named get\_requirements in the setup.py file which will look into the ‘requirements.txt’ file and will install the packages that are mentioned in the requirements.txt

from setuptools import find\_packages,setup

setup(

name = 'MLProject',

version = '0.0.1',

author = 'Pretom Ghosh',

author\_email = 'pretomghoshcuet@gmail.com',

packages = find\_packages(),

install\_requires = get\_requirements('requirements.txt')

from setuptools import find\_packages,setup

from typing import List

def get\_requirements(file\_path:str)->List[str]:

    '''

    this function will the return the list of requirements

    '''

    requirements = []

    with open(file\_path) as file\_obj:

        requirements = file\_obj.readlines()

        requirements = [req.replace("\n","") for req in requirements]

setup(

name = 'MLProject',

version = '0.0.1',

author = 'Pretom Ghosh',

author\_email = 'pretomghoshcuet@gmail.com',

packages = find\_packages(),

install\_requires = get\_requirements('requirements.txt')

)

1. The get\_requirements function is like this:

def get\_requirements(file\_path:str)->List[str]:

    '''

    this function will the return the list of requirements

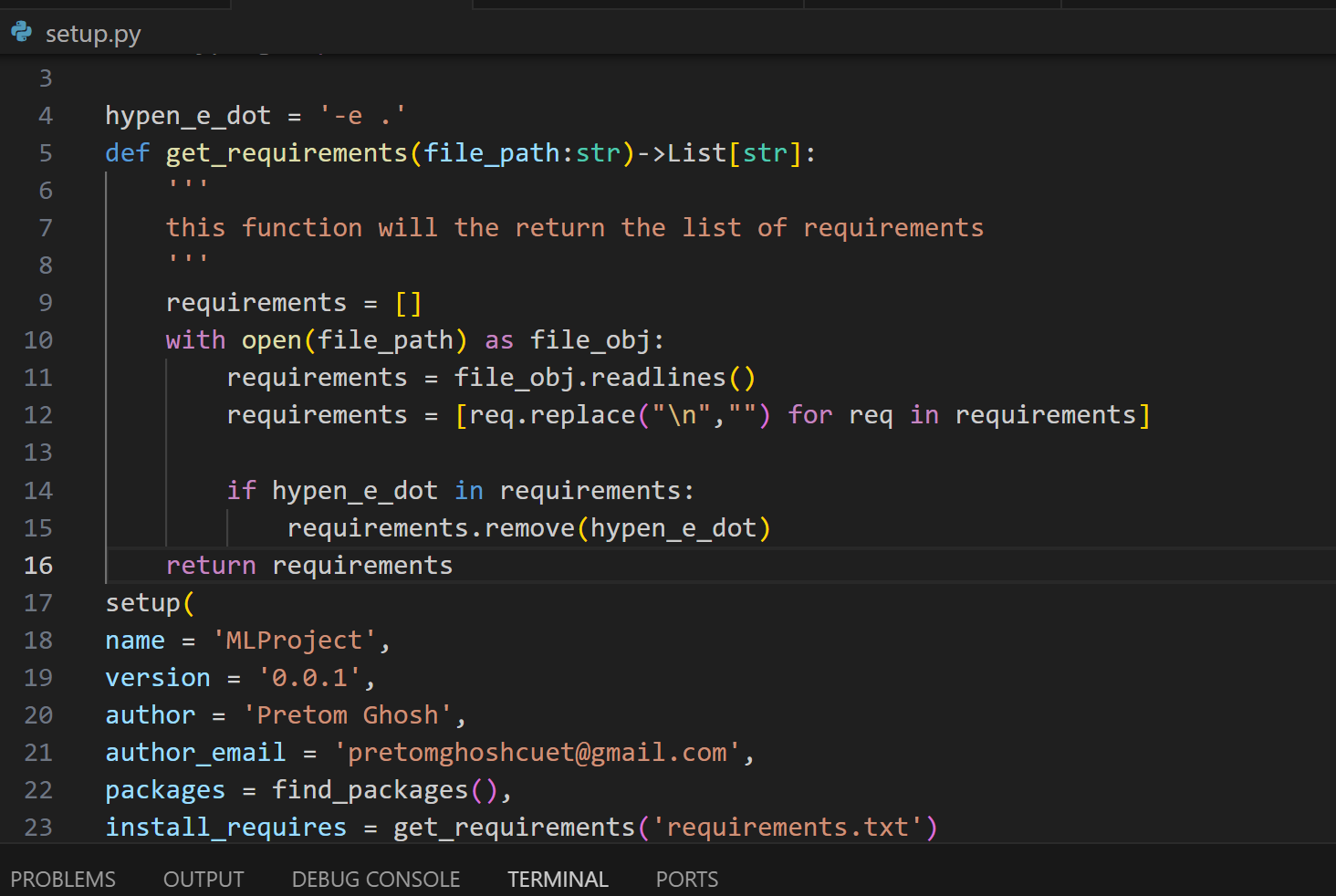
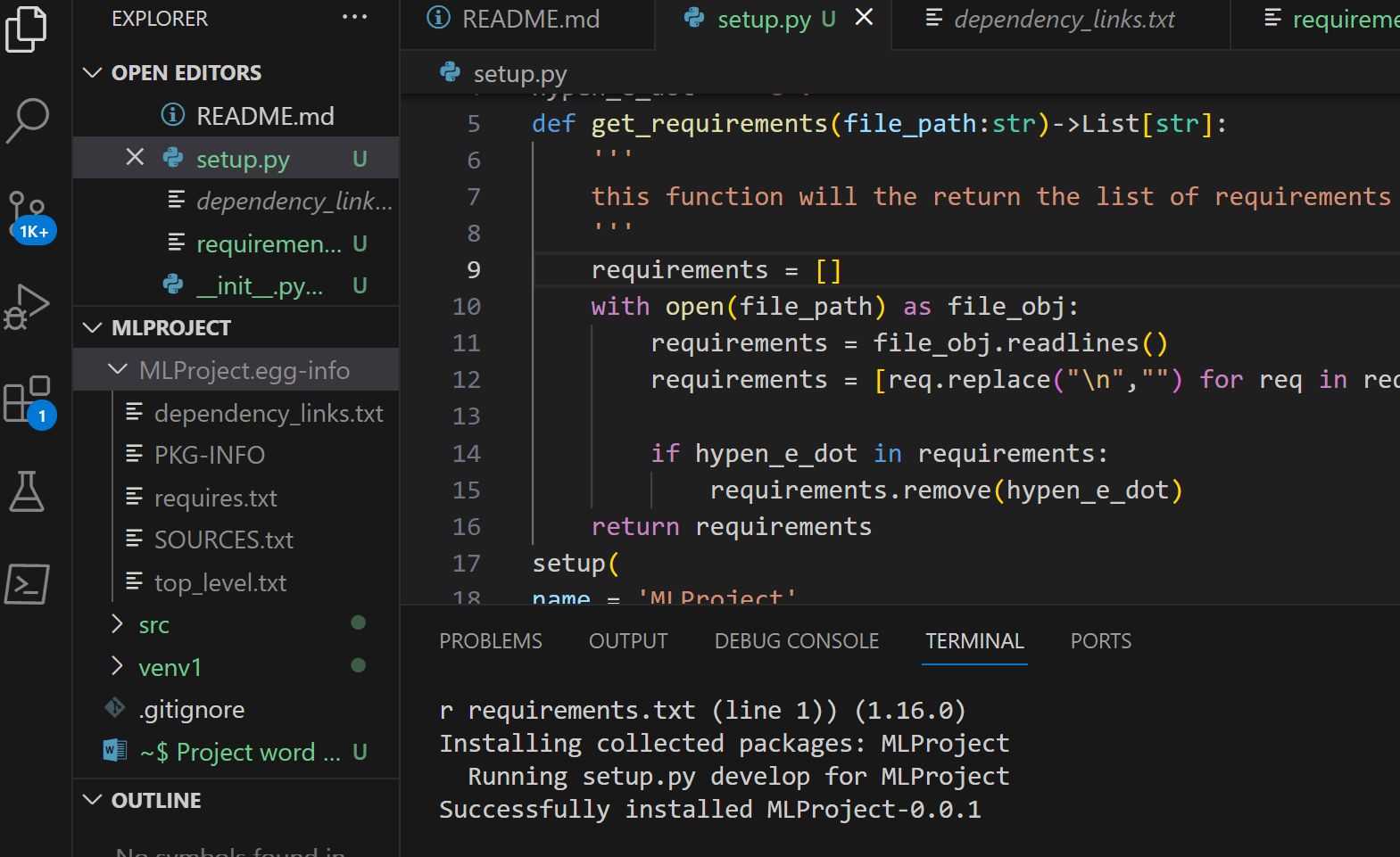
    '''

    requirements = []

    with open(file\_path) as file\_obj:

        requirements = file\_obj.readlines()

        requirements = [req.replace("\n","") for req in requirements]

1. The get\_requirements function will read each lines in the requirements.txt but there will be \n coming in between the lines which we removed by using the replace function.
2. While we install the libraries in the requirements.txt file at the same time setup.py should also run to build the packages. To do that I have used -e . in the requirements.txt
3. At the same time -e . should not interrupt setup.py that’s why I have added an if statement to remove -e . 
4. Now I executed: pip install -r requirements.txt due to which the setup.py ran and created all the required packages and installed the libraries mentioned in the requirements.txt file 

So above what we did are:

1. Set up the github {Repository} by creating

a) new environment

b) setup.py

c) requirements.txt

1. Src folder and build the package
2. We have now created a folder named components under src which later can be used as a package. That’s why we have created a \_\_init\_\_.py file in the components folder
3. In the components folder we have created a data\_ingestion.py file in it which will basically contain the codes required to ingest the data from the databases or the sources. Train and test data split will occur here.
4. Now in the components folder we have created a data\_transformation.py file in it which will basically transform the data.
5. Again, we have created a model\_trainer.py file in the components folder which will basically train the model.
6. We have created a model\_pusher.py file in the components folder which will basically push the model for deployment.
7. Under the src folder we have added a folder named pipeline inside which we have added \_\_init\_.py, train\_pipeline.py and prediction\_pipeline.py files.
8. Now in the src folder I have added logger.py for logging, exception.py for exception handling and added utils.py to add utility functions.
9. In the exception.py I have created a function which will give the details of the error message and print them. And I also created a custom exception which will also print the error message itself. 