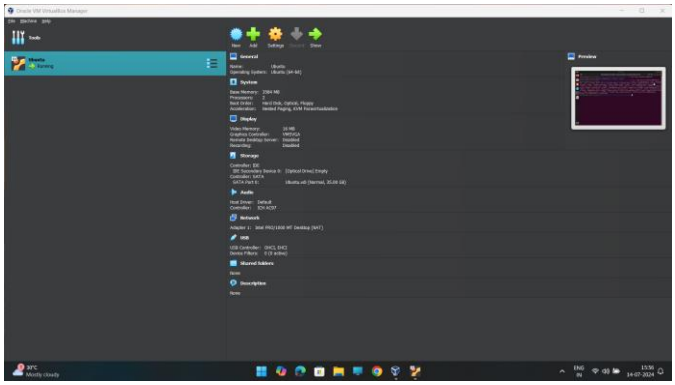
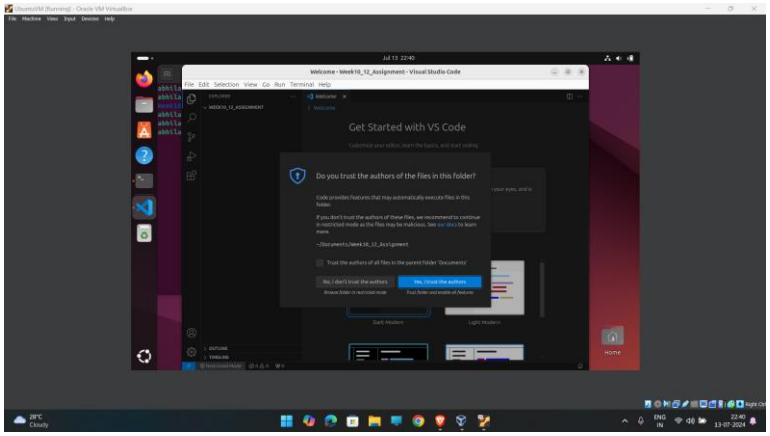
1.Hosted Ubuntu VM using Oracle VM Virtual box

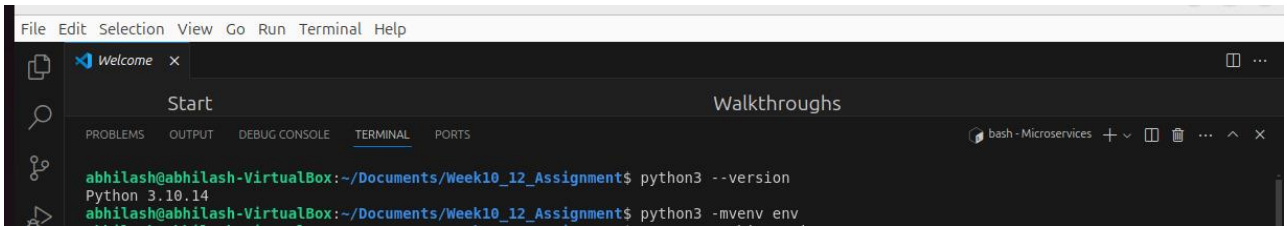


2.Downloaded the VScode file from https://code.visualstudio.com/

Used Terminal to install it.

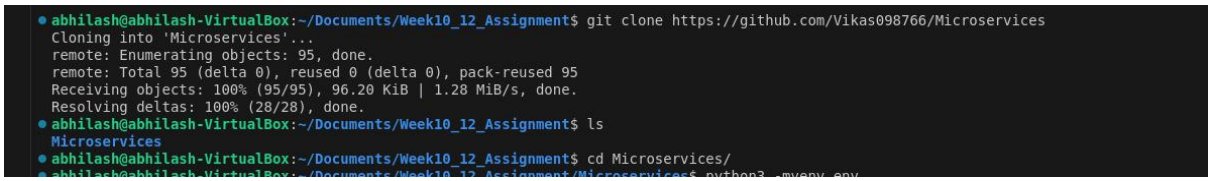


3.Python is set up as the following version



4. Cloned using the command

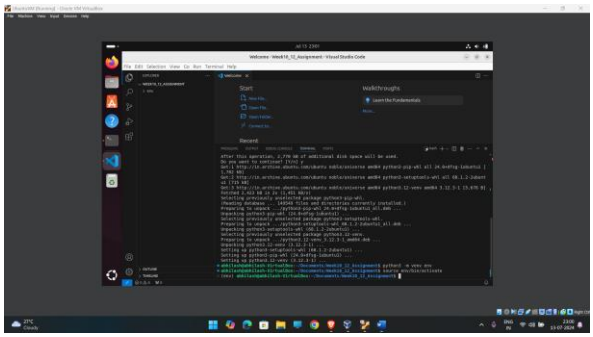
git clone https://github.com/Vikas098766/Microservices.git



5. Created Virtual Environment using commands

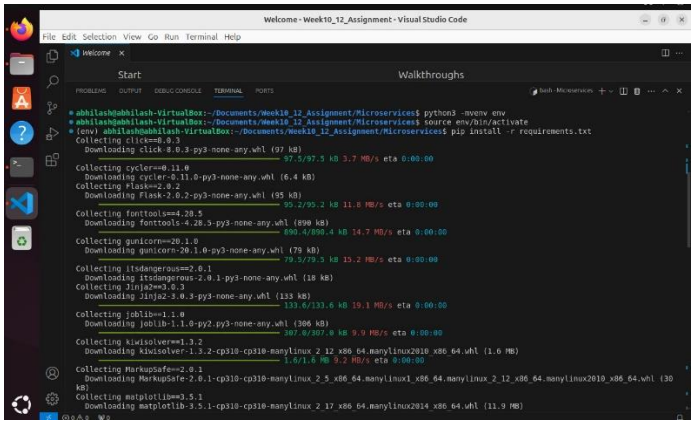
a) python3 -m venv venv

b) source venv/bin/activate



6: Installed all dependencies present in requirements.txt file using the command

Pip install -r requirements.txt



7: Trained and saved the model.

Command: python code_model_training/train.py

```
    Found existing installation: pip 23.0.1
    Uninstalling pip-23.0.1:
      Successfully uninstalled pip-23.0.1
Successfully installed pip-24.1.2
(env) abhilash@abhilash-VirtualBox:~/Documents/Week10_12_Assignment/Microservices$ python code_model_training/train.py
Accuracy: 0.9736842105263158
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x79b593d72200>
/home/abhilash/Documents/Week10_12_Assignment/Microservices/code_model_training/train.py:54: UserWarning: Matplotlib is currently usi
ng agg, which is a non-GUI backend, so cannot show the figure.
  plt.show()
(env) abhilash@abhilash-VirtualBox:~/Documents/Week10_12_Assignment/Microservices$
```

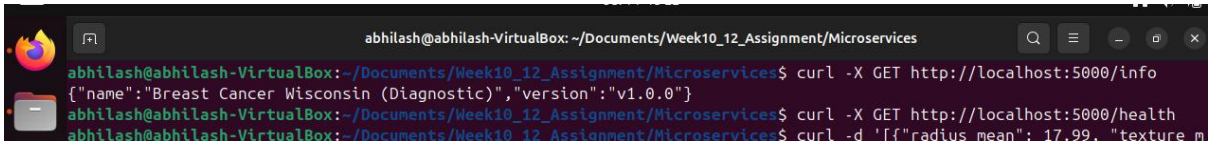
8: Tested web application by running the command. ● flask run -p 5000

```
  plt.show()
(env) abhilash@abhilash-VirtualBox:~/Documents/Week10_12_Assignment/Microservices$ flask run -p 5000
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

9: Tested the the end point /info

Command: curl -X GET http://localhost:5000/info

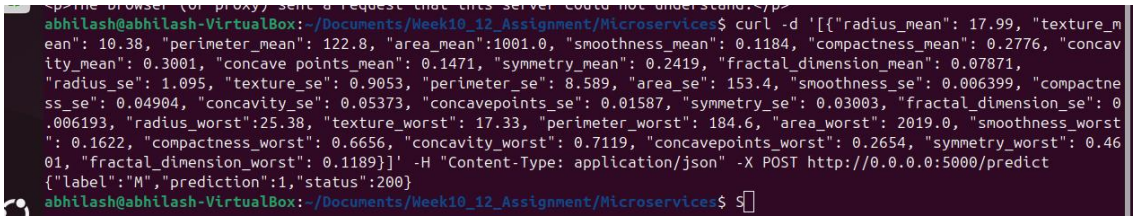
Command: curl -X GET http://localhost:5000/health



Used :
Command : curl -d '[{"radius_mean": 17.99, "texture_mean": 10.38, "perimeter_mean": 122.8, "area_mean": 1001.0, "smoothness_mean": 0.1184, "compactness_mean": 0.2776, "concavity_mean": 0.3001, "concave points_mean": 0.1471, "symmetry_mean": 0.2419, "fractal_dimension_mean": 0.07871, "radius_se": 1.095, "texture_se": 0.9053, "perimeter_se": 8.589, "area_se": 153.4, "smoothness_se": 0.006399, "compactness_se": 0.04904, "concavity_se": 0.05373, "concave points_se": 0.01587, "symmetry_se": 0.03003, "fractal_dimension_se": 0.006193, "radius_worst": 25.38, "texture_worst": 17.33, "perimeter_worst": 184.6, "area_worst": 2019.0, "smoothness_worst": 0.1622, "compactness_worst": 0.6656, "concavity_worst": 0.7119, "concave points_worst": 0.2654, "symmetry_worst": 0.4601, "fractal_dimension_worst": 0.1189}]'

\ -H "Content-Type: application/json"

\-X POST http://0.0.0.0:5000/predict

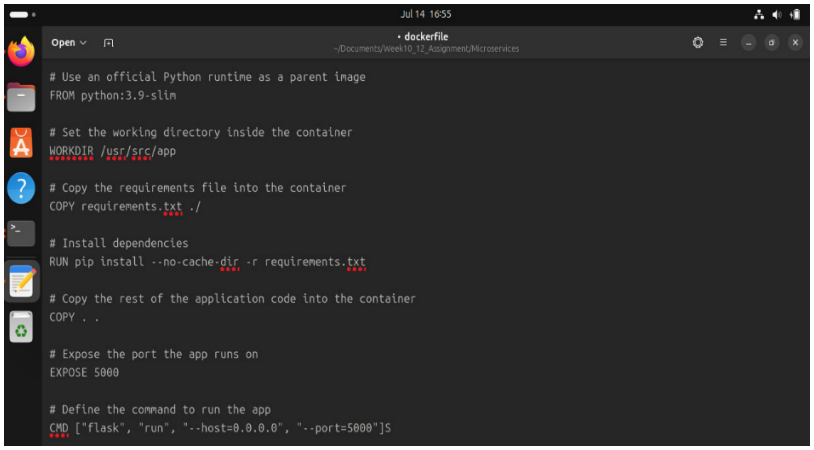


API ENDPOINT with /predict got the output as {“label”:”M”,” prediction”:1,” status”:200}

10. Steps to create a docker image.

1. Created the text file named dockerfile using the command as touch dockerfile

2. Within the txt file adding the following content within it.



3. Build the docker image with the name as my-python-app

Command: sudo docker build -t my-python-app .

```
abhilash@abhilash-VirtualBox:~/Documents/Week10_12_Assignment/Microservices$ sudo docker build -t my-python-app .
[+] Building 57.8s (10/10) FINISHED                                                                    docker:default
 => [internal] load build definition from dockerfile                                                           0.0s
 => => transferring dockerfile: 539B                                                                           0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                             2.7s
 => [internal] load .dockerignore                                                                              0.0s
 => => transferring context: 2B                                                                                0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:a6c12ec09f13df9d4b8b4e4d08678c1b212d89885be14b6c72b633be  10.9s
 => => resolve docker.io/library/python:3.9-slim@sha256:a6c12ec09f13df9d4b8b4e4d08678c1b212d89885be14b6c72b633bee  0.0s
 => => sha256:c1f67e58a3d2a9d9c5f38c8c3fc629ff3bfd6e0045b935c99e9ffc4182070fa1 3.51MB / 3.51MB                 3.1s
 => => sha256:9370038d11852cad5a70691e76b0ddc8e669018bc770cad15c23a3def629b874 11.89MB / 11.89MB               3.7s
 => => sha256:a6c12ec09f13df9d4b8b4e4d08678c1b212d89885be14b6c72b633bee2a520f4 10.41kB / 10.41kB               0.0s
 => => sha256:4719115deb9cc7a5479a7d3c57cfceac2be89fcaf0fed8c747e8dfb4b01a79a3 1.94kB / 1.94kB                 0.0s
 => => sha256:b97320a8c1caf64deeebb911ff8eb75bf12f671408a85302dd33b5ede2d1cdc1 6.90kB / 6.90kB                 0.0s
 => => sha256:f11c1adaa26e078479ccdd45312ea3b88476441b91be0ec898a7e07bfd05badc 29.13MB / 29.13MB               4.5s
 => => sha256:174cb52e37e55d01d6ea95cdffa00d252ccd883946a96050ff4be1633a7f1712 231B / 231B                     3.6s
 => => sha256:f259657f3656970a8d44c5a7a8250b746967f869516bb767767c6942eaa60e55 2.77MB / 2.77MB                 4.3s
 => => extracting sha256:f11c1adaa26e078479ccdd45312ea3b88476441b91be0ec898a7e07bfd05badc                      2.3s
 => => extracting sha256:c1f67e58a3d2a9d9c5f38c8c3fc629ff3bfd6e0045b935c99e9ffc4182070fa1                      0.5s
 => => extracting sha256:9370038d11852cad5a70691e76b0ddc8e669018bc770cad15c23a3def629b874                      1.3s
 => => extracting sha256:174cb52e37e55d01d6ea95cdffa00d252ccd883946a96050ff4be1633a7f1712                      0.0s
 => => extracting sha256:f259657f3656970a8d44c5a7a8250b746967f869516bb767767c6942eaa60e55                      0.5s
 => [internal] load build context                                                                             10.8s
 => => transferring context: 449.67MB                                                                         10.7s
 => [2/5] WORKDIR /usr/src/app                                                                                 0.4s
 => [3/5] COPY requirements.txt ./                                                                             0.0s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt                                                  28.6s
 => [5/5] COPY . .                                                                                             7.9s
 => exporting to image                                                                                         7.2s
 => => exporting layers                                                                                        7.1s
 => => writing image sha256:3b2958c5e1d4adee66ac4ee491f5f86fc4dc7843e8f482630b00fccbfd34ad7d                   0.0s
 => => naming to docker.io/library/my-python-app                                                               0.0s
```

4. Run the Docker Container

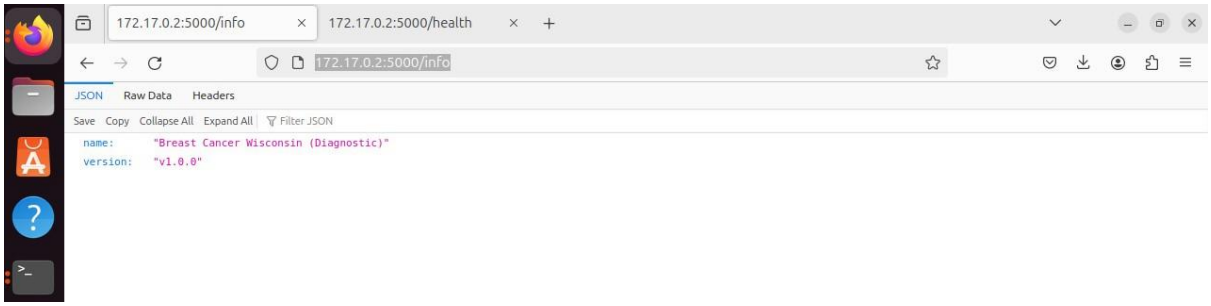
Command: sudo docker run -p 5000:5000 my-python-app

Docker image is run successfully.

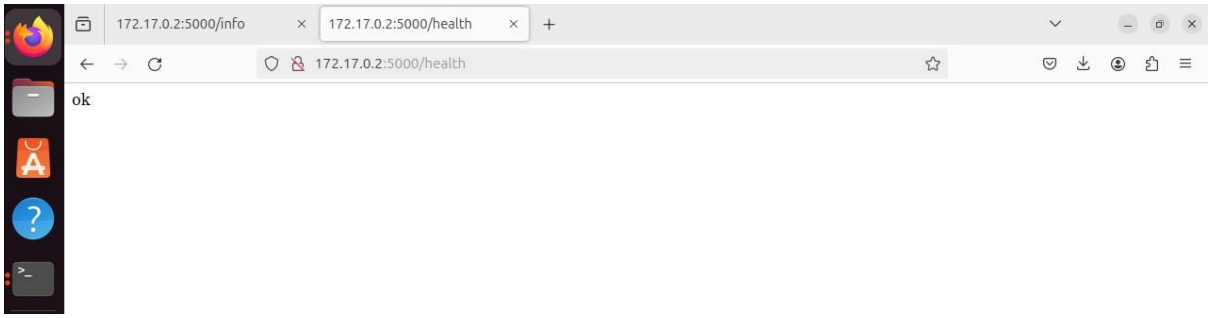
```
abhilash@abhilash-VirtualBox:~/Documents/Week10_12_Assignment/Microservices$ sudo docker run -p 5000:5000 my-python-app
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
```

11. To check the Docker image service locally with the help of POSTMAN end points as

1. /info



2. /health



3. /predict