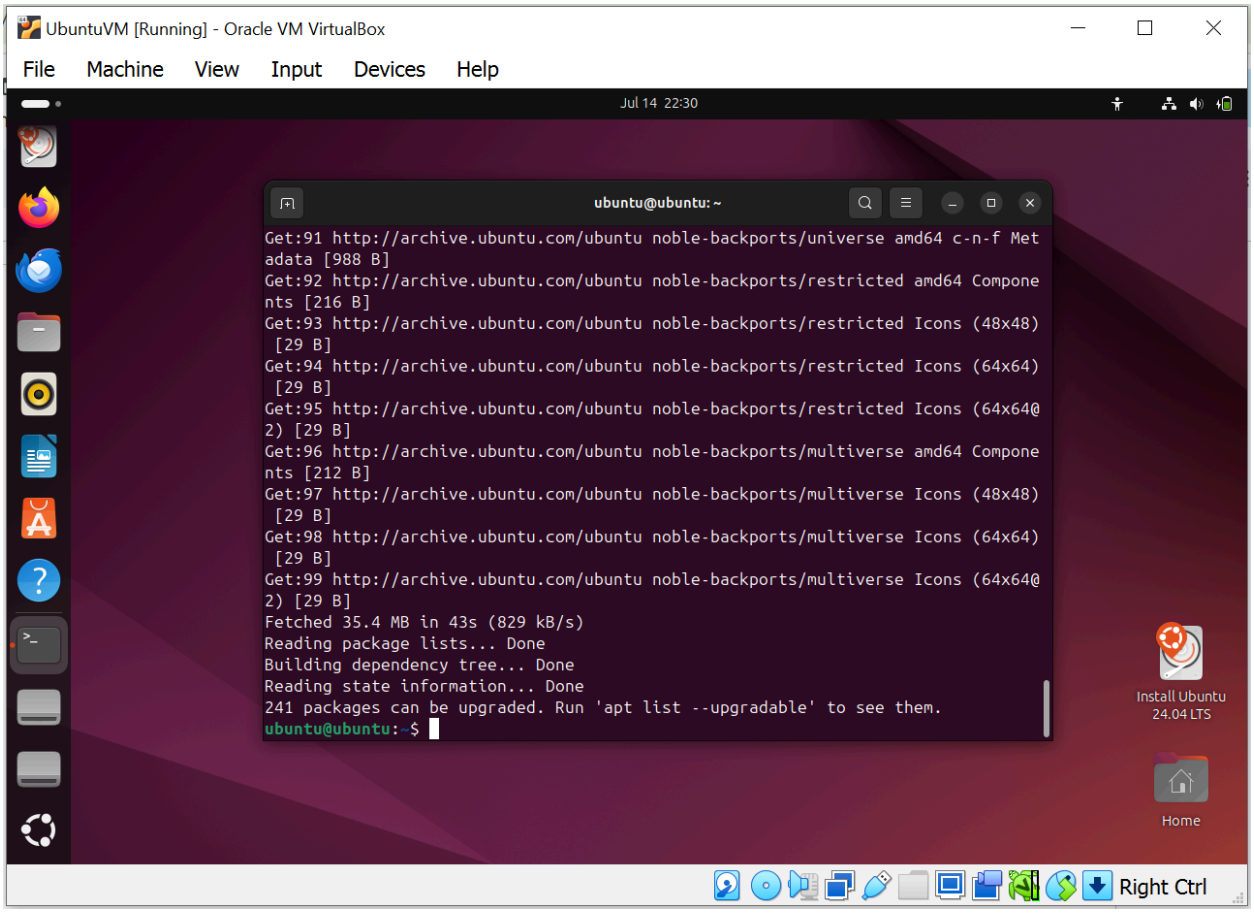
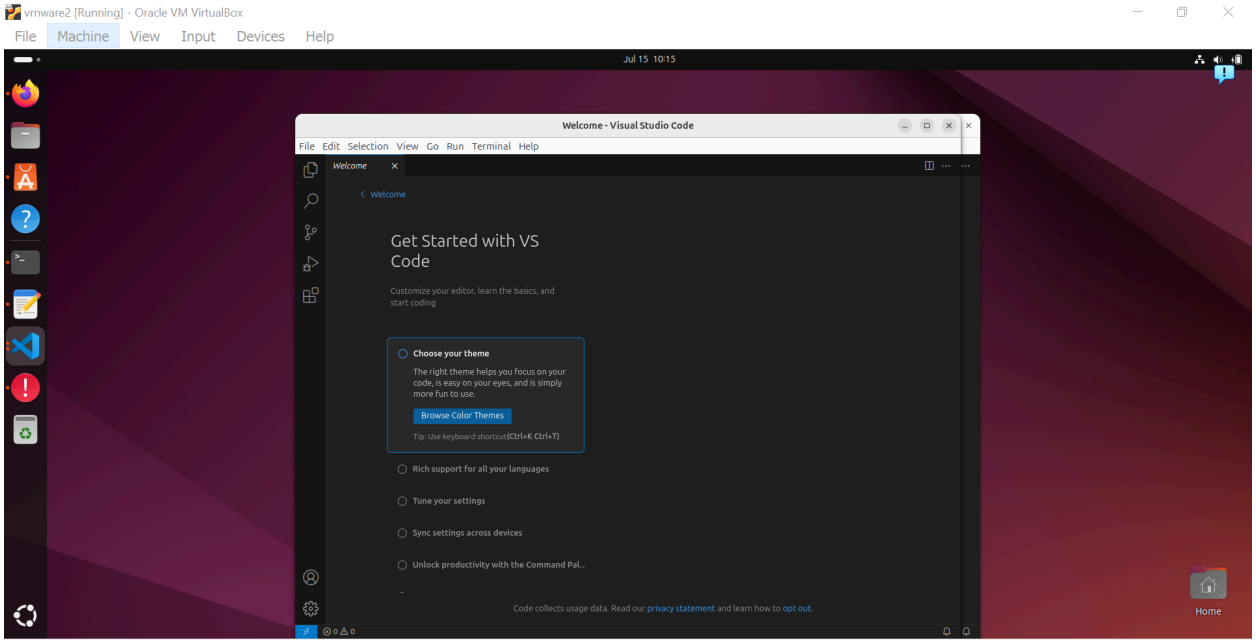
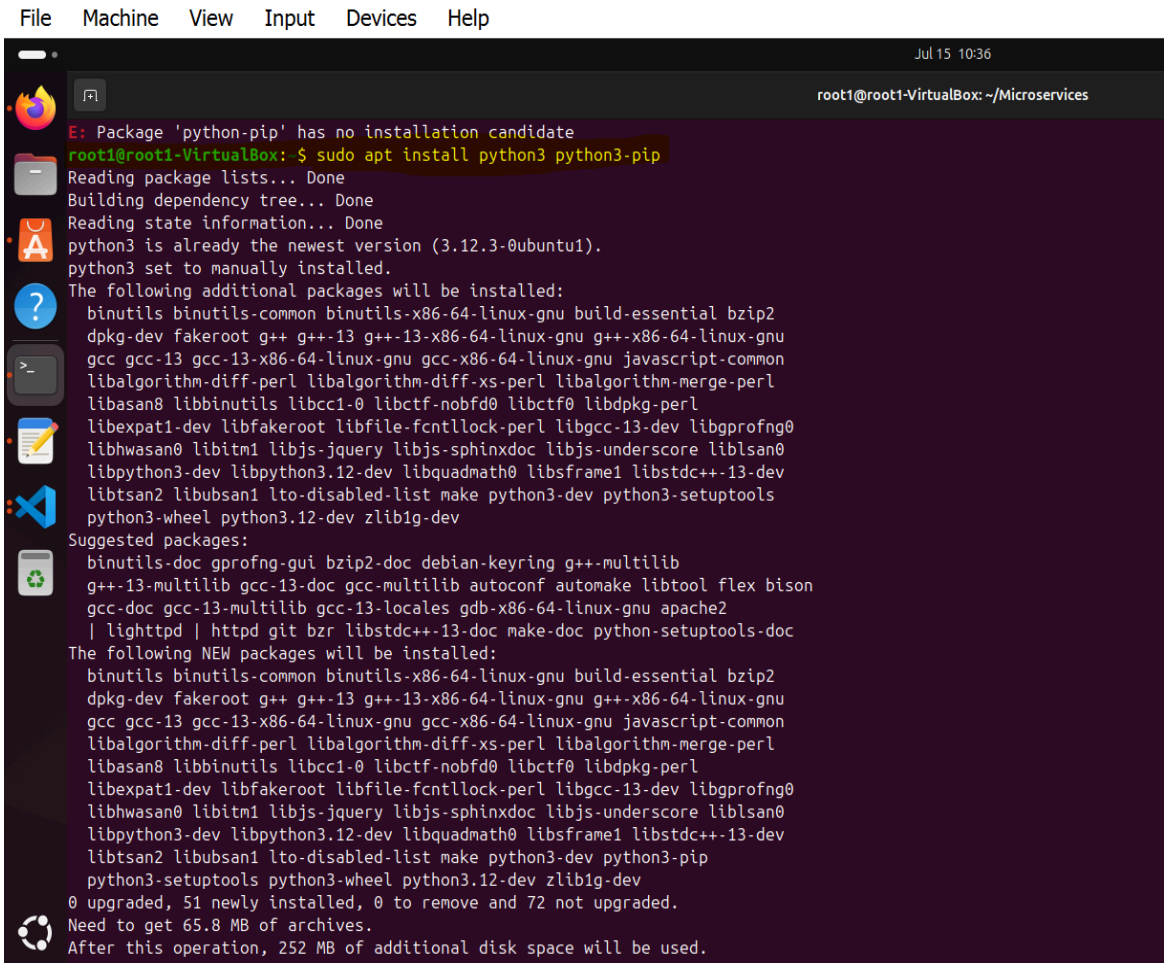
1. Host a Ubuntu Virtual Machine using Oracle VM Virtual Box.



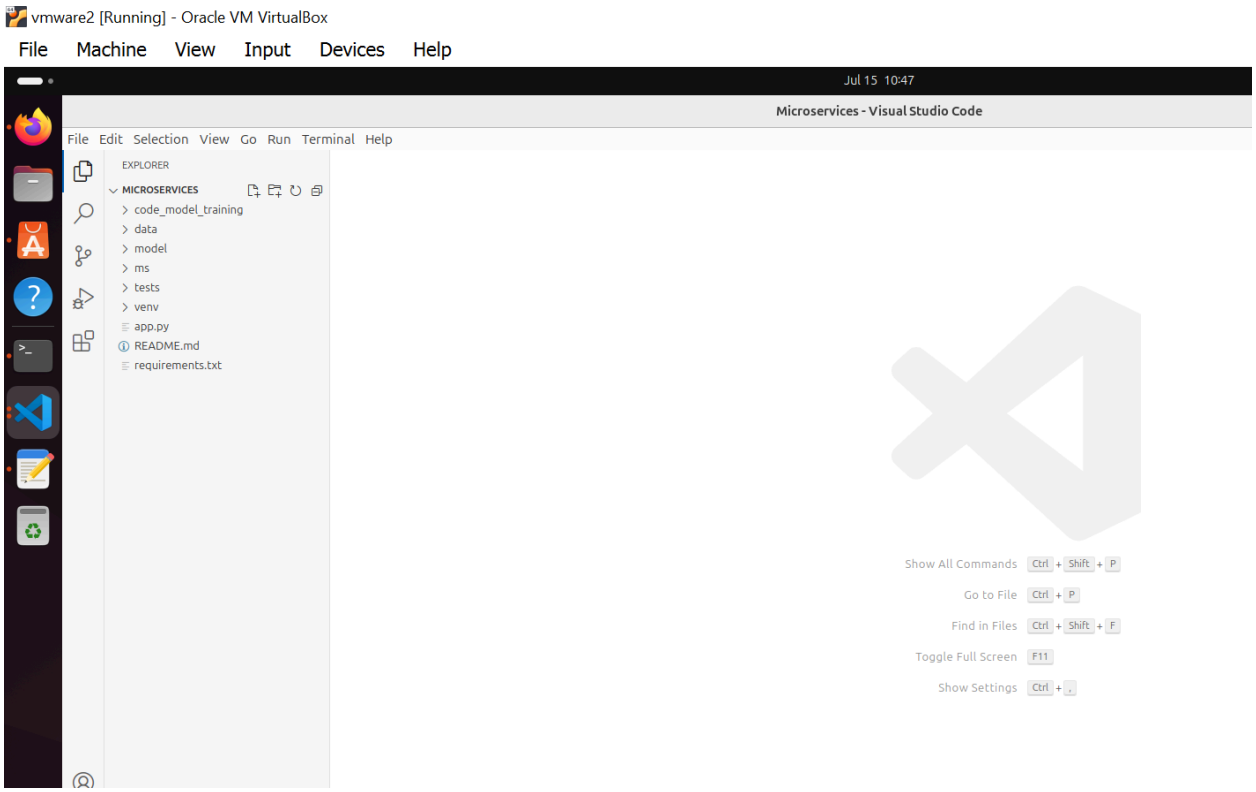

2. Set up Visual Studio code on Ubuntu VM.



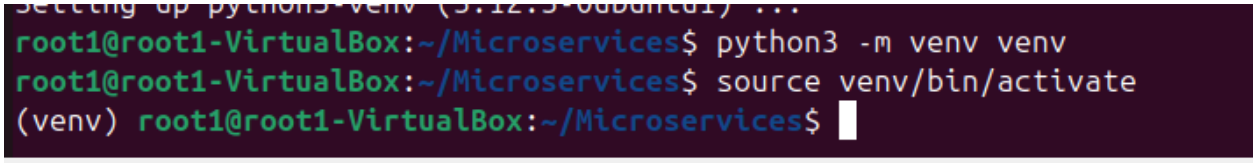
3. Set up Python




4. Clone this Github repository https://github.com/Vikas098766/Microservices.git




5. Create a Virtual Environment.

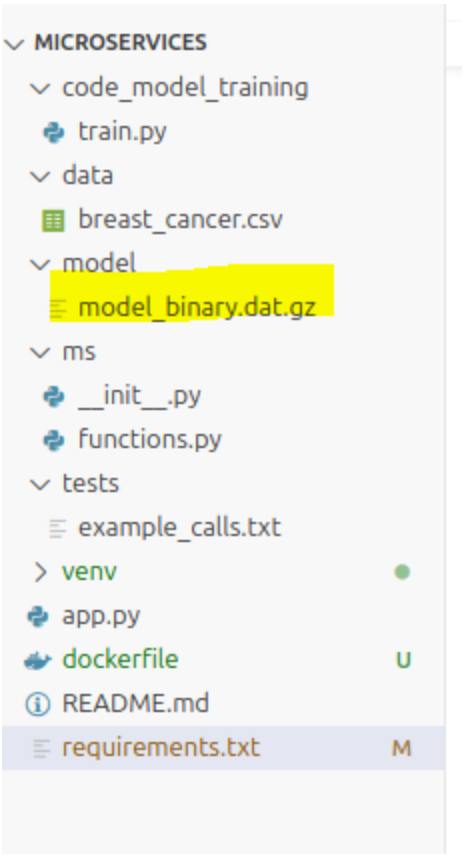



6. Install the dependencies from requirements.txt file.

```
(venv) root1@root1-VirtualBox:~/Microservices$ pip install -r requirements.txt
Collecting click==8.0.3 (from -r requirements.txt (line 1))
  Downloading click-8.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting cycler==0.11.0 (from -r requirements.txt (line 2))
  Downloading cycler-0.11.0-py3-none-any.whl.metadata (785 bytes)
Collecting Flask==2.0.2 (from -r requirements.txt (line 3))
  Downloading Flask-2.0.2-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools==4.28.5 (from -r requirements.txt (line 4))
  Downloading fonttools-4.28.5-py3-none-any.whl.metadata (118 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 119.0/119.0 kB 891.6 kB/s eta 0:00:00
Collecting gunicorn==20.1.0 (from -r requirements.txt (line 5))
  Downloading gunicorn-20.1.0-py3-none-any.whl.metadata (3.8 kB)
Collecting itsdangerous==2.0.1 (from -r requirements.txt (line 6))
  Downloading itsdangerous-2.0.1-py3-none-any.whl.metadata (2.9 kB)
Collecting Jinja2==3.0.3 (from -r requirements.txt (line 7))
  Downloading Jinja2-3.0.3-py3-none-any.whl.metadata (3.5 kB)
Collecting joblib==1.1.0 (from -r requirements.txt (line 8))
  Downloading joblib-1.1.0-py2.py3-none-any.whl.metadata (5.2 kB)
Collecting kiwisolver==1.3.2 (from -r requirements.txt (line 9))
  Downloading kiwisolver-1.3.2.tar.gz (54 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.6/54.6 kB 989.4 kB/s eta 0:00:00
  Installing build dependencies ... done
```

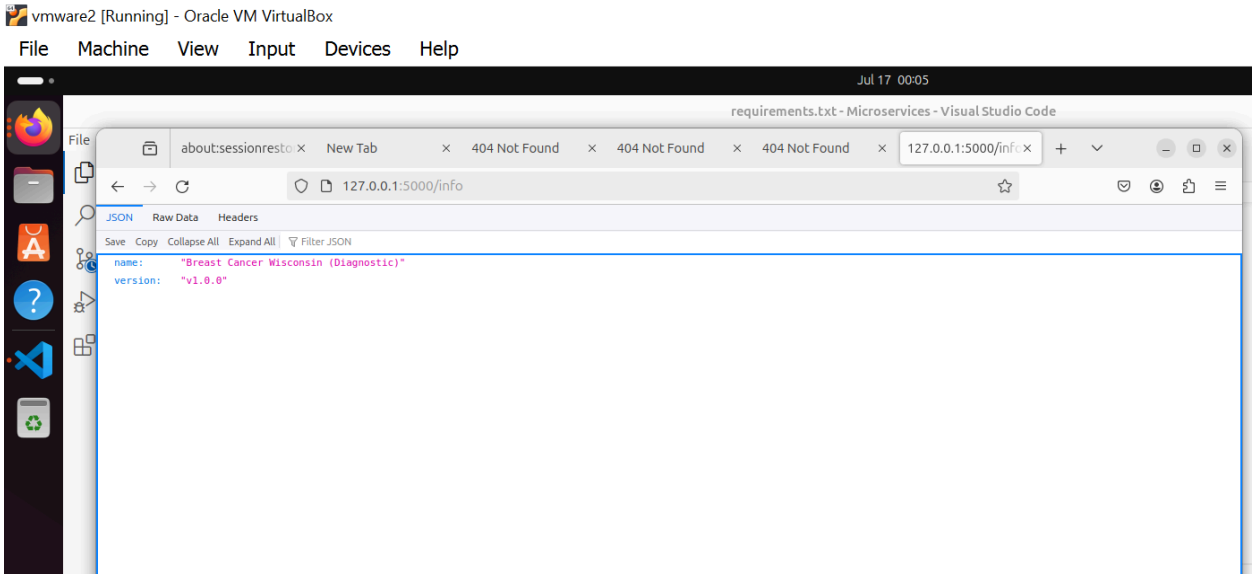
7. Train and save the model.

```
(venv) root1@root1-VirtualBox:~/Microservices$ python code_model_training/train.py
/home/root1/Microservices/code_model_training/train.py:23: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To re
tain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  data['diagnosis'] = data['diagnosis'].replace(['B', 'M'], [0, 1])  # Encode y, B -> 0 , M -> 1
Accuracy: 0.9736842105263158
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x73b26f5ab1a0>
/home/root1/Microservices/code_model_training/train.py:54: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  plt.show()
```




8. Test the Flask web application.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(venv) root1@root1-VirtualBox:~/Microservices$ python app.py
 * Serving Flask app 'ms'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
```



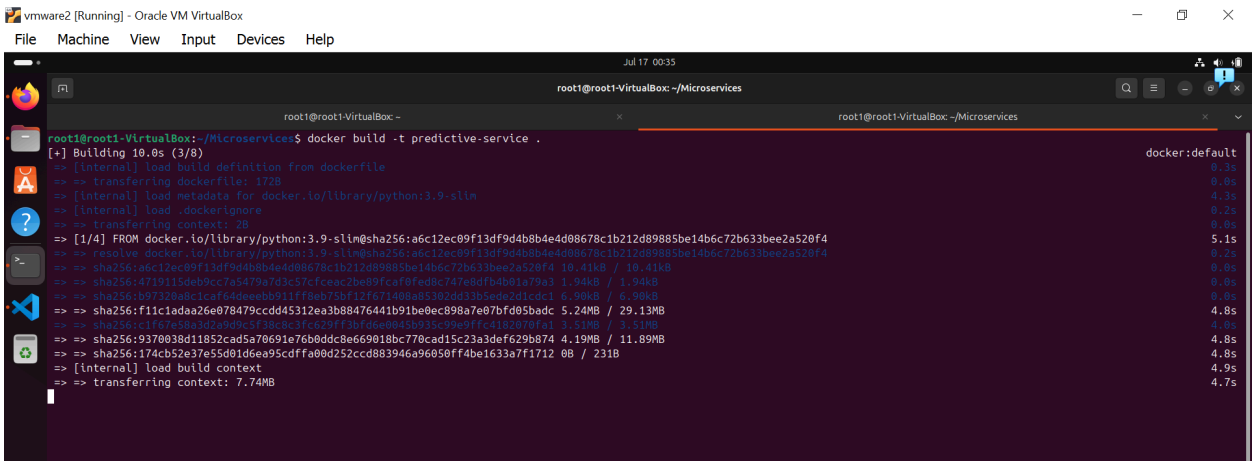

9. Test the application and make predictions using the example calls available in the folder/tests.


10. Create a docker image containing everything needed to run the application.

requirements.txt M | example_calls.txt | dockerfile U

dockerfile

```
2   WORKDIR /app
3   COPY . /app
4   RUN pip install -r requirements.txt
5   EXPOSE 5000
6   ENV NAME World
7   CMD ["python", "app.py"]
```




11. Run the containerized application as a prediction service and test it locally by passing some example calls and get the prediction.