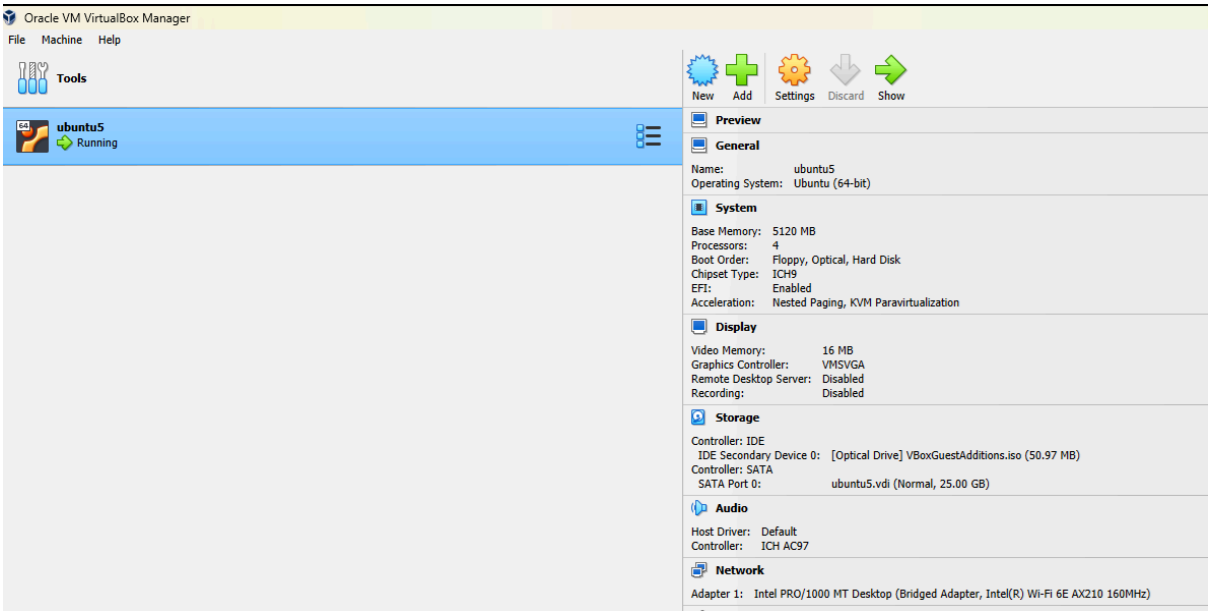
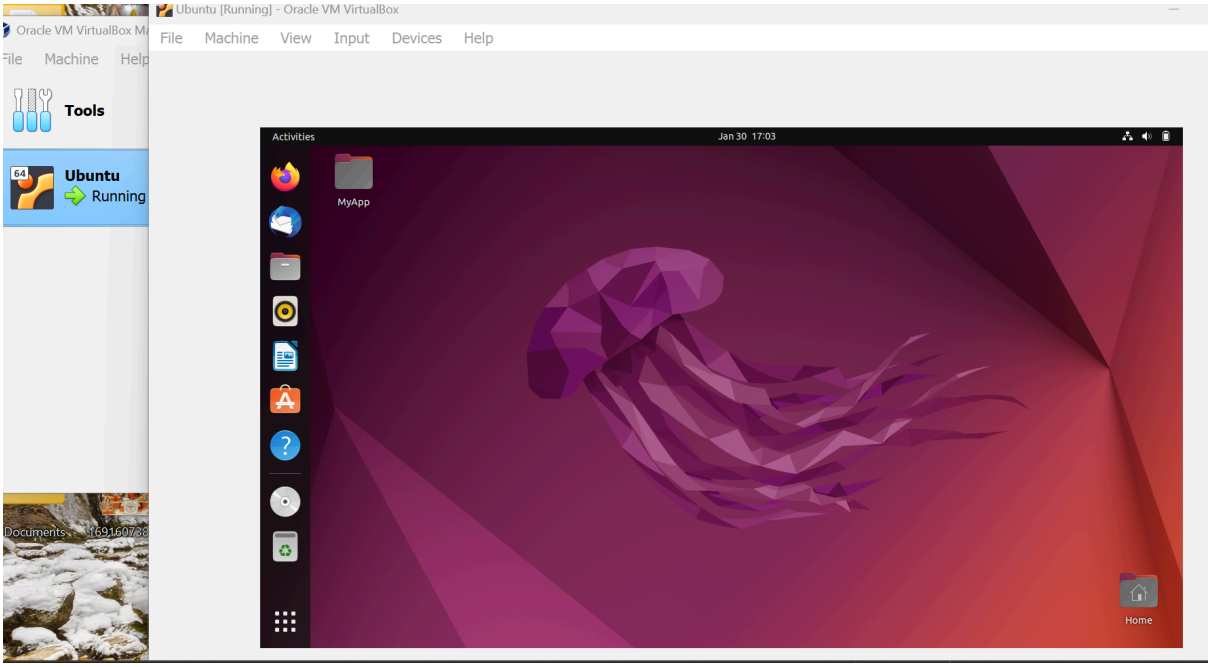
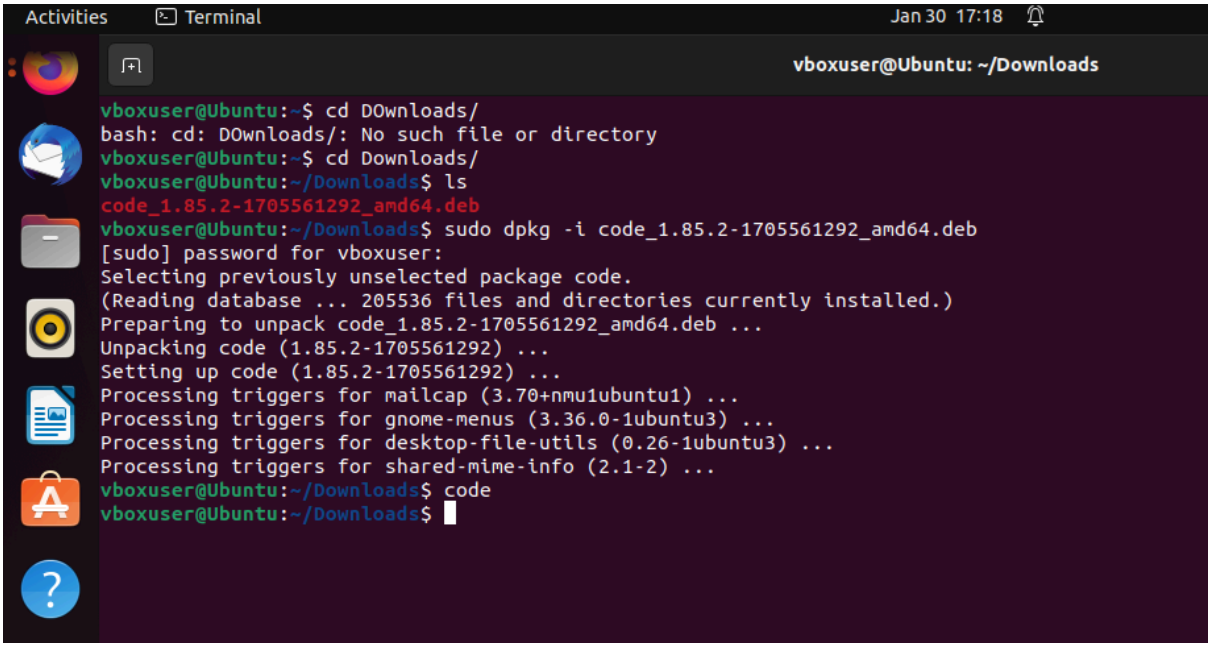
Steps performed, commands for that specific task, and screenshots for each step

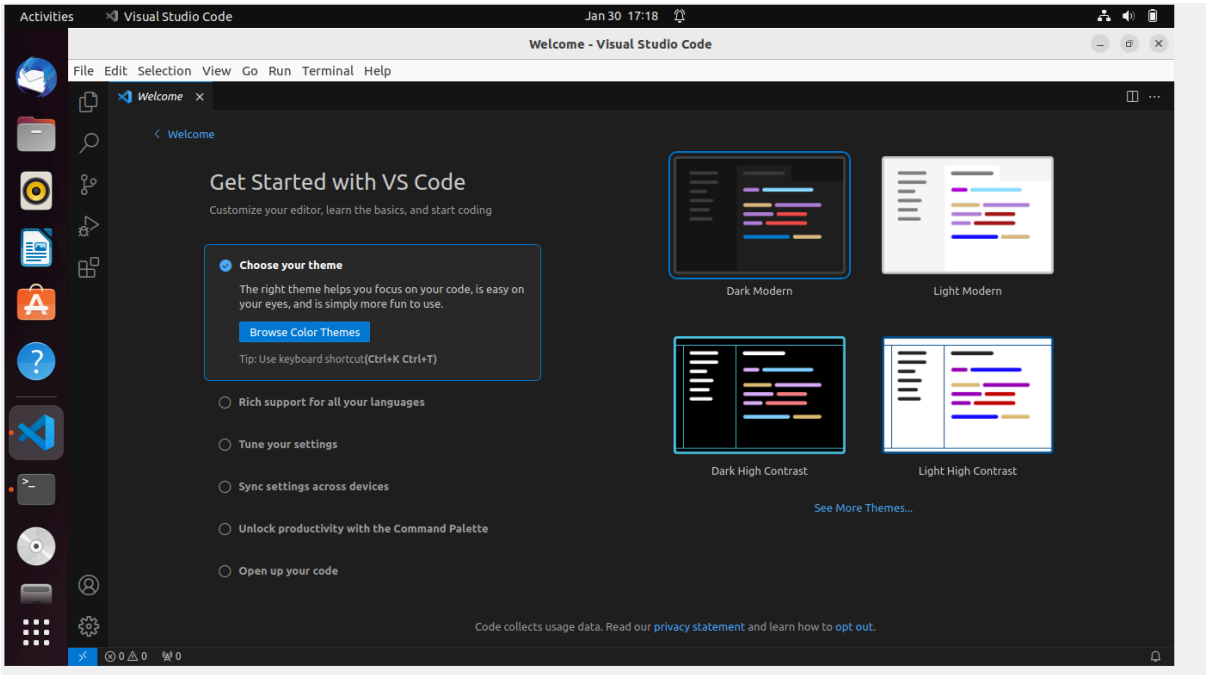
1. Host a Ubuntu Virtual Machine using Oracle VM Virtual Box. (5 marks)



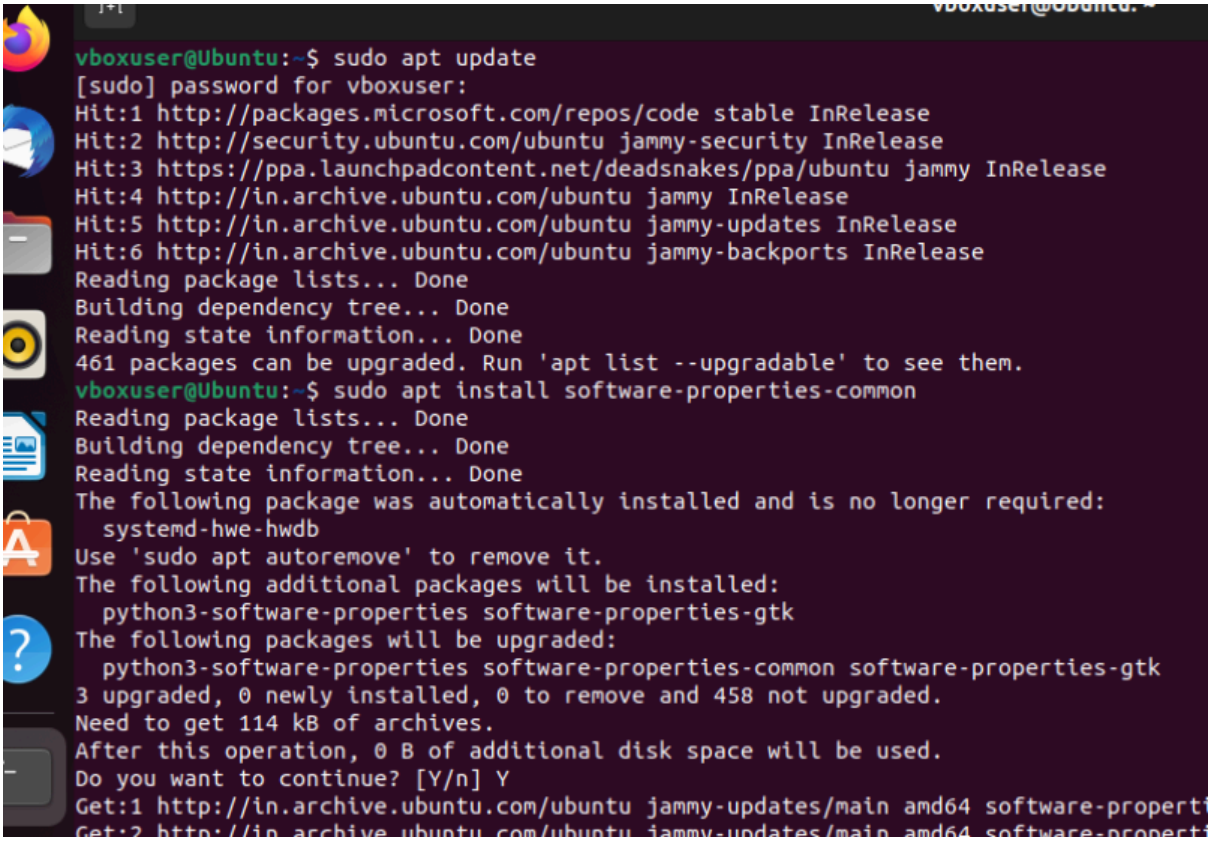


2. Set up Visual Studio code on Ubuntu VM. (5 marks)

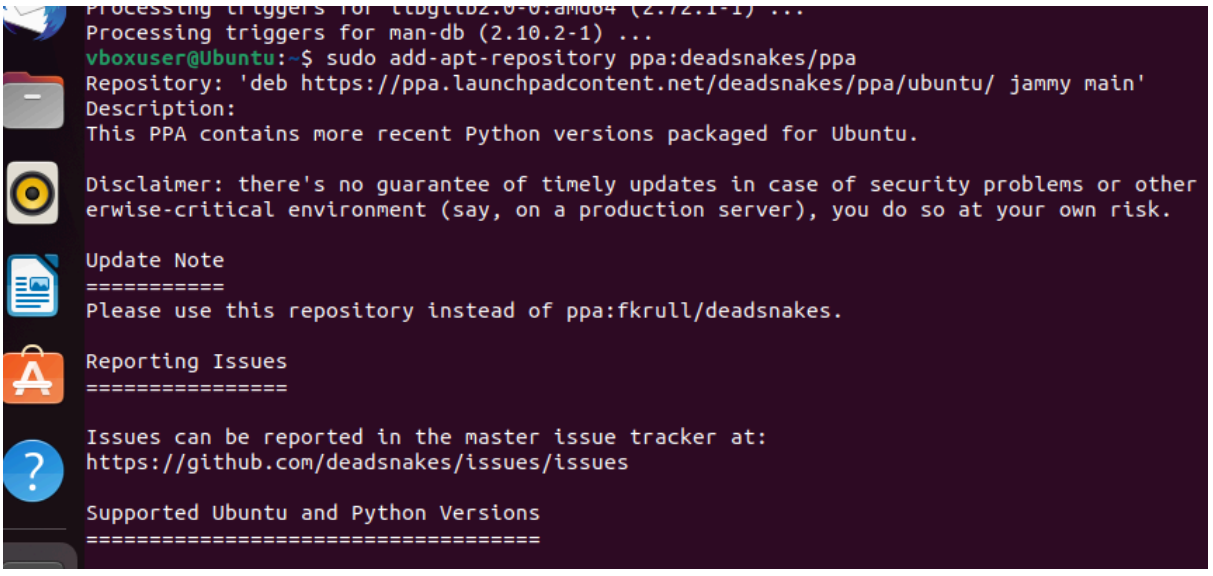




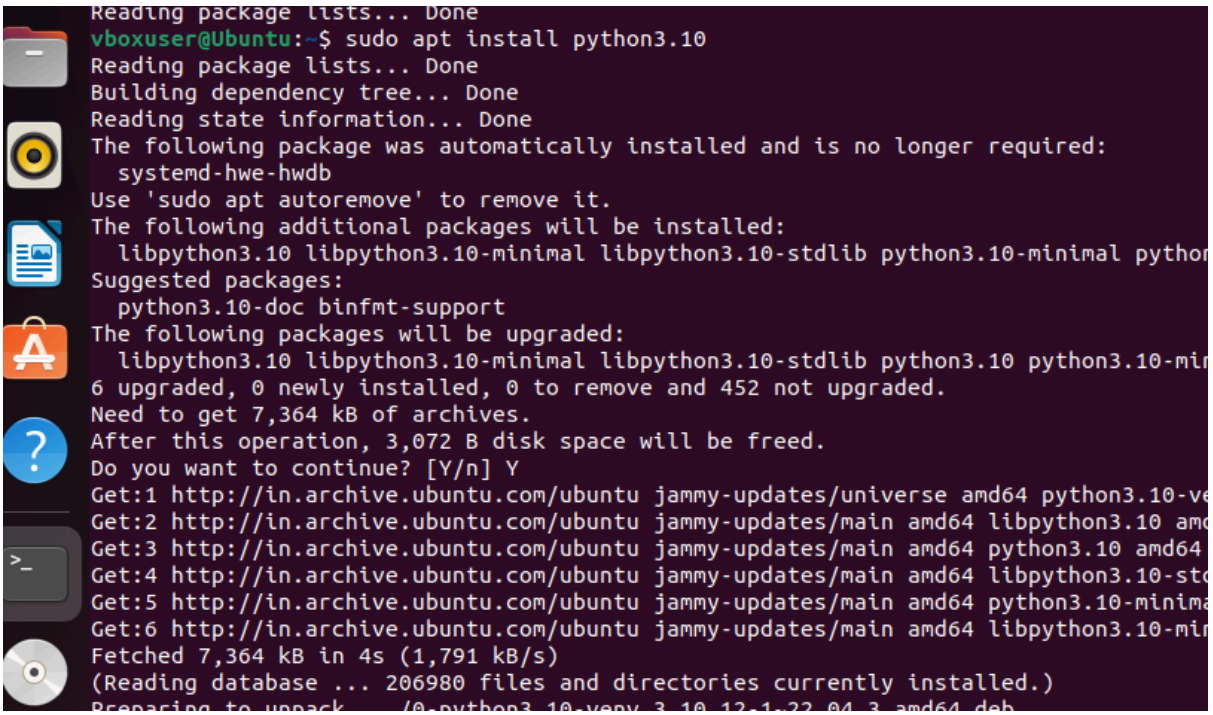
3. Set up Python. (5 marks)

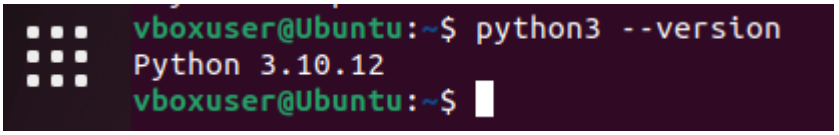


```
vboxuser@Ubuntu:~$ sudo apt update
[sudo] password for vboxuser:
Hit:1 http://packages.microsoft.com/repos/code stable InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
461 packages can be upgraded. Run 'apt list --upgradable' to see them.
vboxuser@Ubuntu:~$ sudo apt install software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  python3-software-properties software-properties-gtk
The following packages will be upgraded:
  python3-software-properties software-properties-common software-properties-gtk
3 upgraded, 0 newly installed, 0 to remove and 458 not upgraded.
Need to get 114 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 software-properti
```

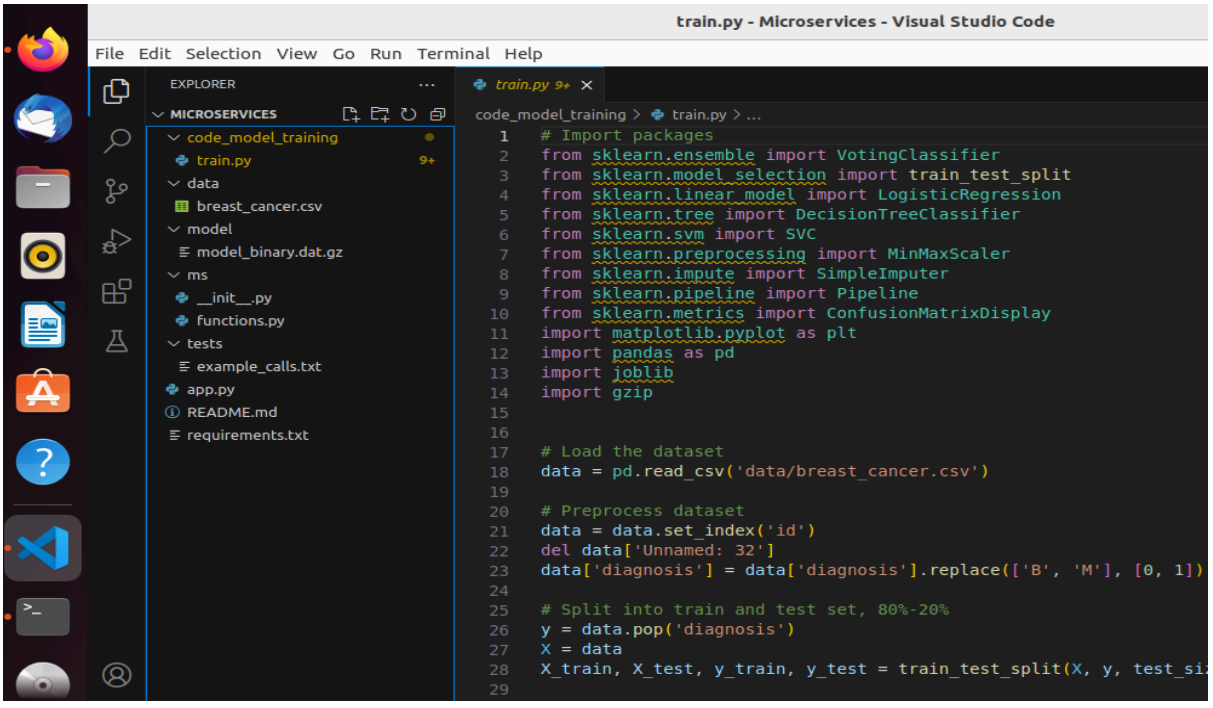


```
Processing triggers for man-db (2.10.2-1) ...
vboxuser@Ubuntu:~$ sudo add-apt-repository ppa:deadsnakes/ppa
Repository: 'deb https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu/ jammy main'
Description:
This PPA contains more recent Python versions packaged for Ubuntu.

Disclaimer: there's no guarantee of timely updates in case of security problems or other
erwise-critical environment (say, on a production server), you do so at your own risk.

Update Note
===========
Please use this repository instead of ppa:fkrull/deadsnakes.

Reporting Issues
================

Issues can be reported in the master issue tracker at:
https://github.com/deadsnakes/issues/issues

Supported Ubuntu and Python Versions
====================================
```

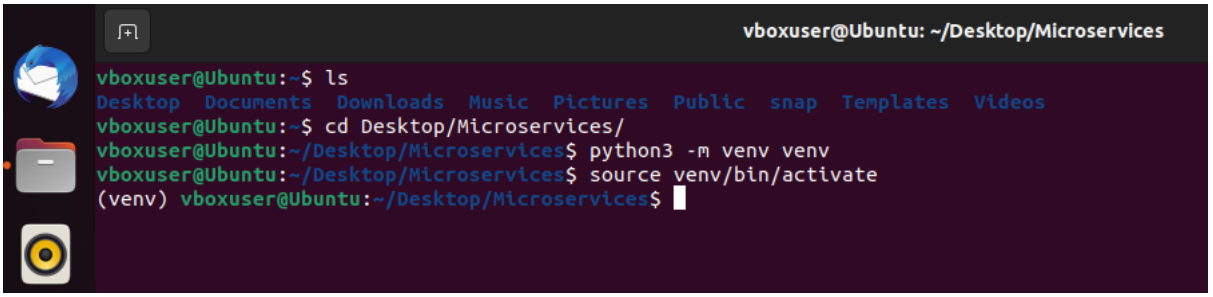




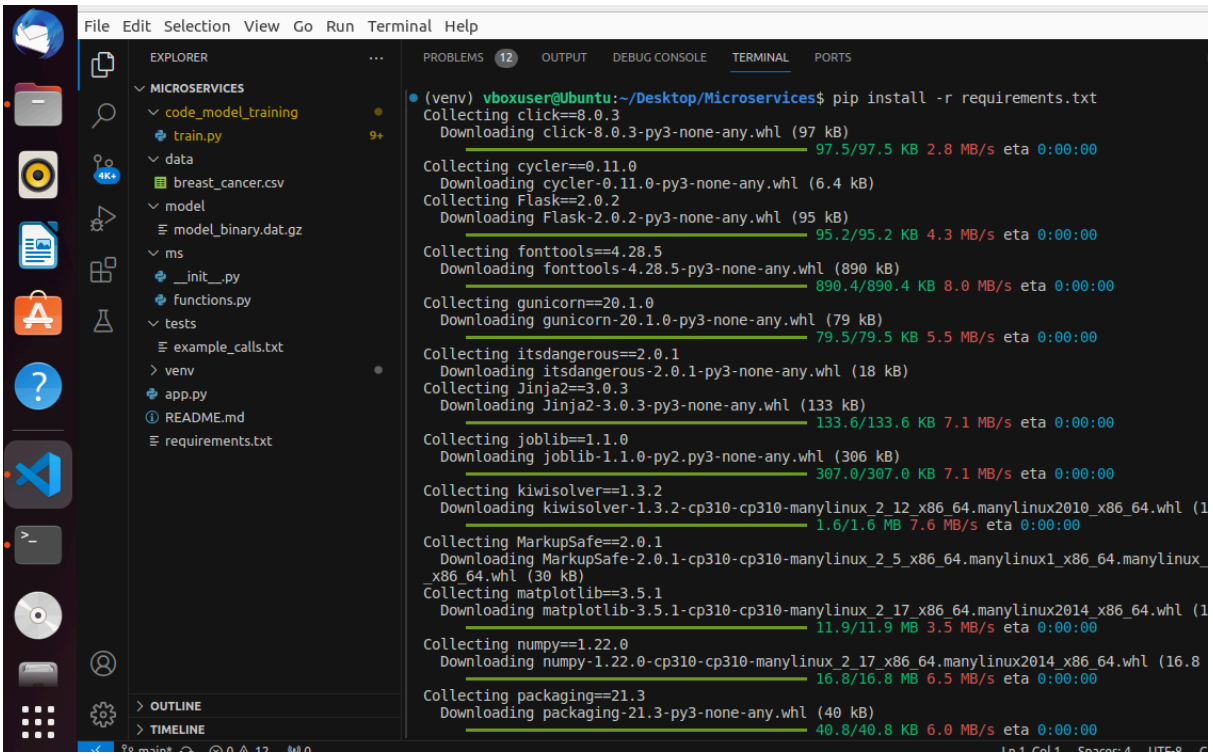
4. Clone this Github repository https://github.com/Vikas098766/Microservices.git (1 mark)

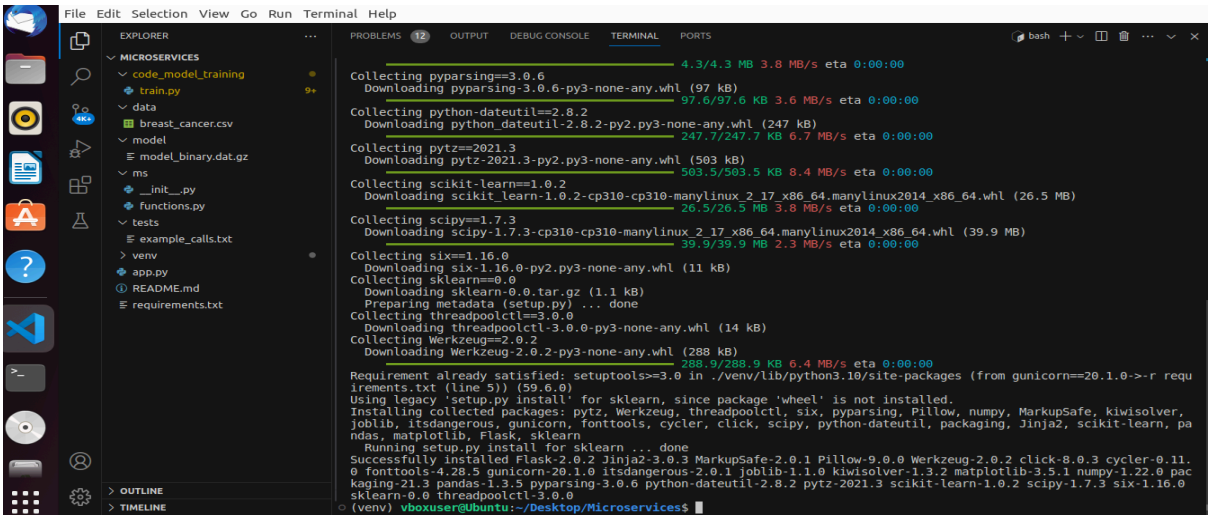


5. Create a Virtual Environment. (1 mark)

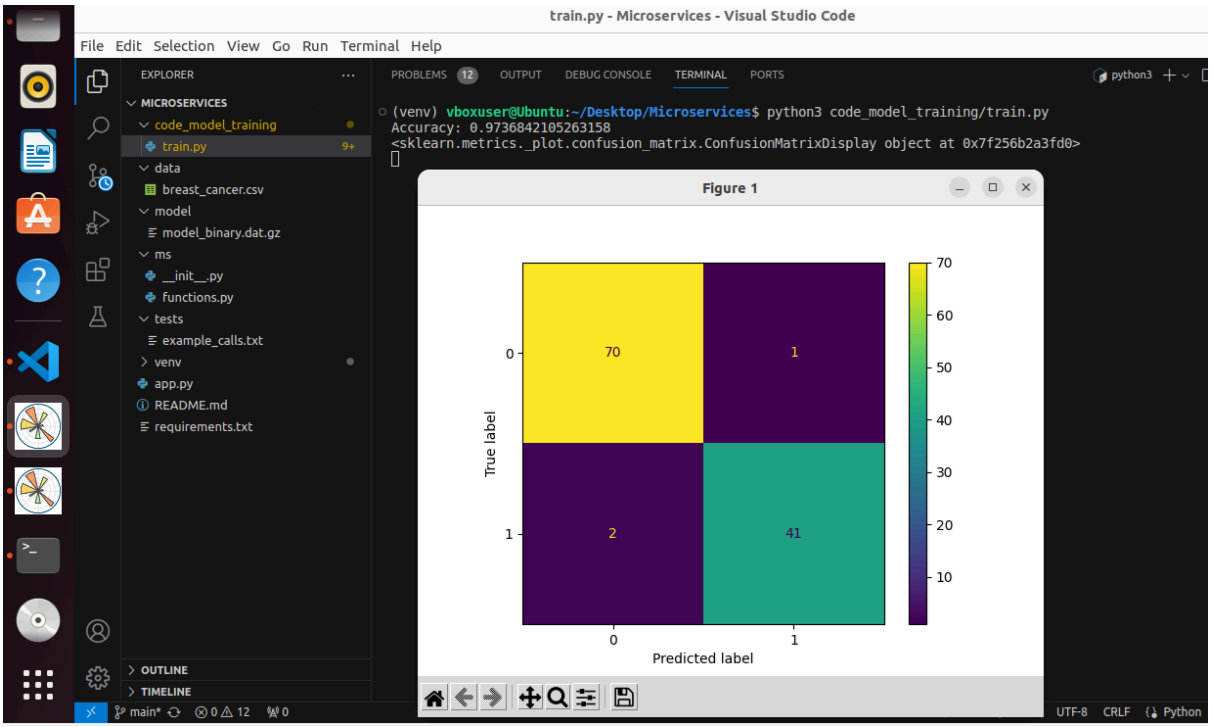


6. Install the dependencies from requirements.txt file. (1 mark)

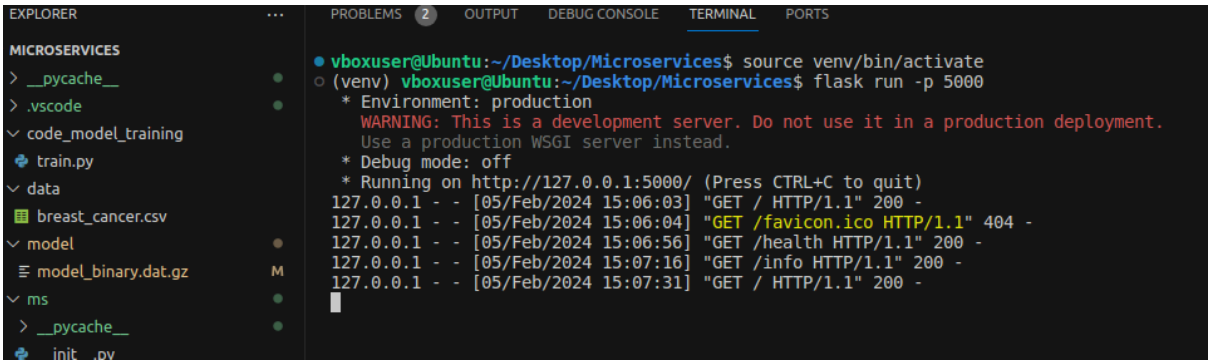




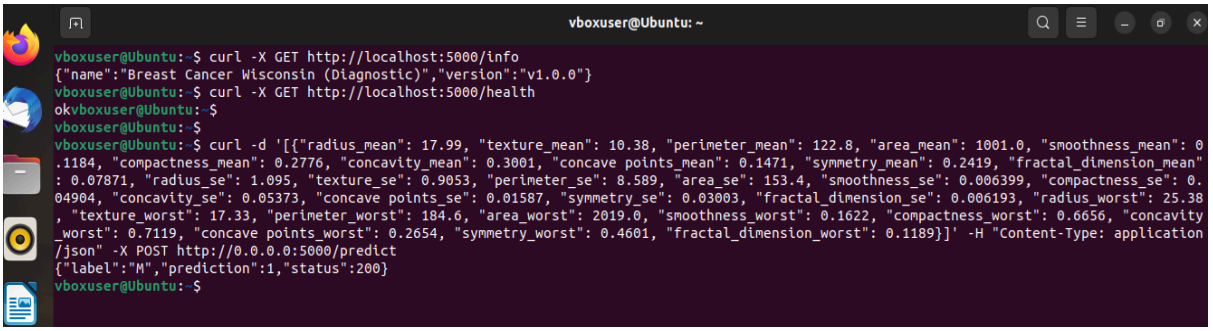
7. Train and save the model. (2 marks)



8. Test the Flask web application. (5 marks)

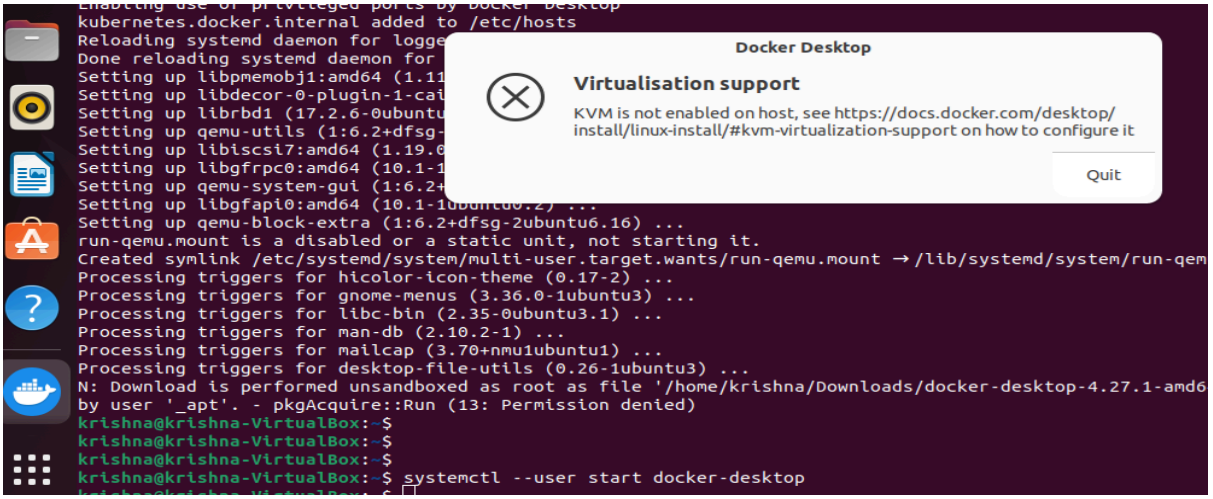


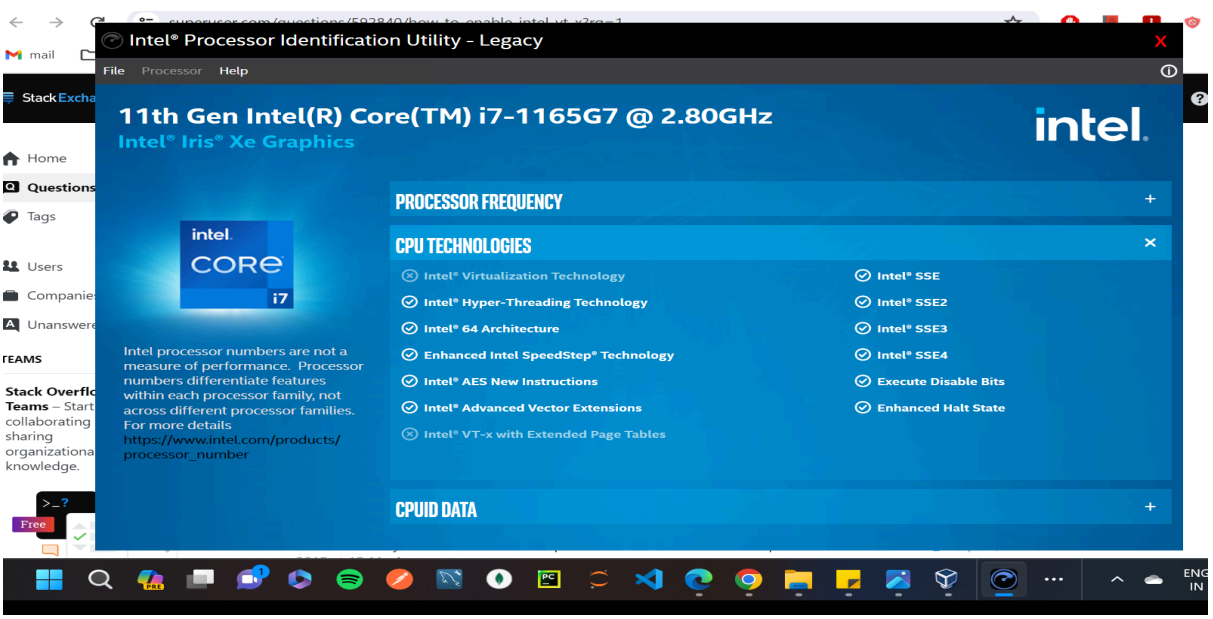
9. Test the application and make predictions using the example calls available in the folder /tests. (5 marks)



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

Note: My PC was not enabling KVM as its hardware doesn't support Virtualization/VT-x/Extended Page Tables.





Alternatively: Hence I did the docker part (10th and 11th tasks) in the Windows environment on my PC by using Docker Desktop.

Dockerfile U compose.yaml U

compose.yaml

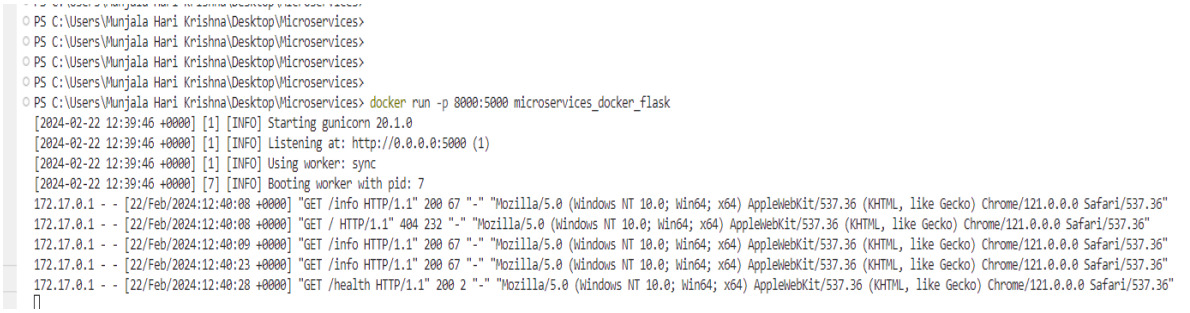
```yaml
# Comments are provided throughout this file to help you get started.
# If you need more help, visit the Docker compose reference guide at
# https://docs.docker.com/go/compose-spec-reference/

# Here the instructions define your application as a service called "server".
# This service is built from the Dockerfile in the current directory.
# You can add other services your application may depend on here, such as a
# database or a cache. For examples, see the Awesome Compose repository:
# https://github.com/docker/awesome-compose
services:
  server:
    build:
      context: .
    ports:
      - 5000:5000

# The commented out section below is an example of how to define a PostgreSQL
# database that your application can use. `depends_on` tells Docker Compose to
# start the database before your application. The `db-data` volume persists the
# database data between container restarts. The `db-password` secret is used
# to set the database password. You must create `db/password.txt` and add
# a password of your choosing to it before running `docker compose up`.
#     depends_on:
#       db:
#         condition: service_healthy
#   db:
#     image: postgres
#     restart: always
#     user: postgres
#     secrets:
#       - db-password
#     volumes:
#       - db-data:/var/lib/postgresql/data
```

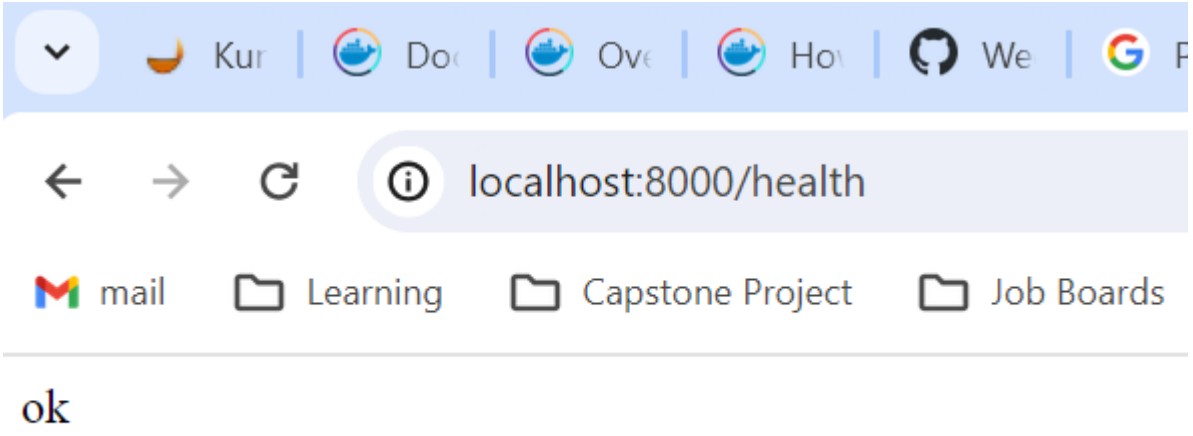
Dockerfile U compose.yaml U

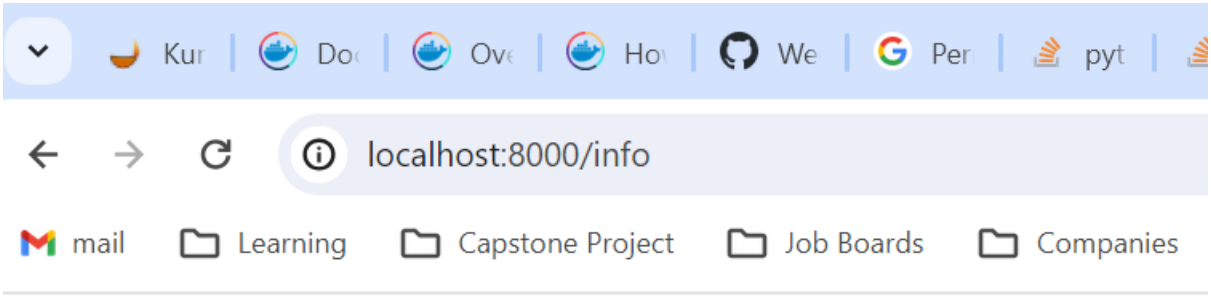
Dockerfile > ...

```dockerfile

# Comments are provided throughout this file to help you get started.
# If you need more help, visit the Dockerfile reference guide at
# https://docs.docker.com/go/dockerfile-reference/

# Want to help us make this template better? Share your feedback here: https://forms.gle/ybq9Krt8jtBL3iCk7

# Base image
FROM python:3.10

# Set working directory
WORKDIR /app

# Copy files
COPY app.py /app
COPY requirements.txt /app
COPY model /app/model
COPY ms /app/ms

# Install dependencies
RUN pip3 install -r requirements.txt

COPY . /app

# Expose the port that the application listens on.
EXPOSE 5000

# Run the application.
ENTRYPOINT ["gunicorn", "-b", "0.0.0.0:5000", "--access-logfile", "-", "--error-logfile", "-", "--timeout", "120"]
CMD ["app:app"]
# RUN python3 ./model/code_model_training/train.py
# CMD python3 ./app.py --host 0.0.0.0
```

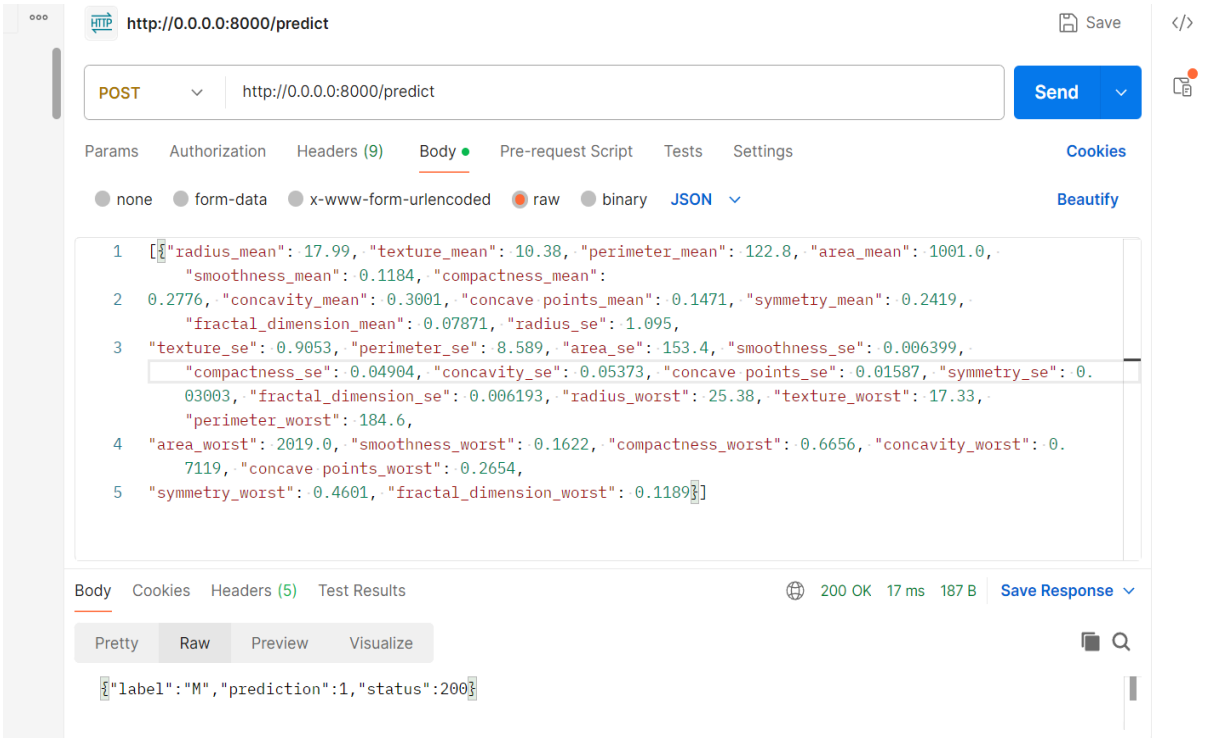
11.Run the containerized application as a prediction service and test it locally by passing some example calls and get the prediction. (10 marks)







{"name":"Breast Cancer Wisconsin (Diagnostic)","version":"v1.0.0"}



Note: All the commands for each task are in the attached screenshots itself.