**PART 2:Graded Assignment**

**Objective:**

Implementing a microservice using the Python Flask framework on an Ubuntu virtual machine to serve a machine learning prediction model.

To create a Docker image containing everything needed to run the application: the application code, libraries, tools, dependencies, and other files and to use the image to run the application in containers.

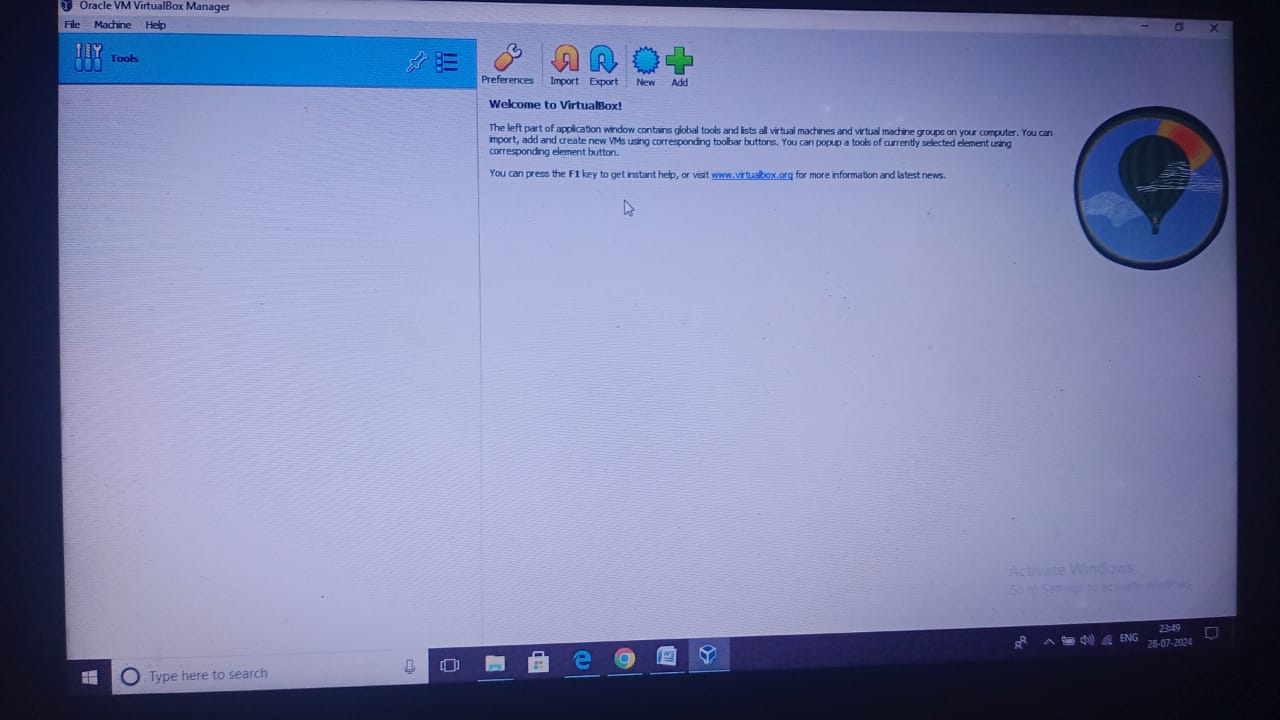
**STEPS PERFORMED For PART 2:**

1. Host a Ubuntu Virtual Machine using Oracle VM Virtual Box. (5 marks)
2. Oracle VM VirtualBox installation -

* Open link for Oracle VM Virtual Box as :

<https://www.virtualbox.org/>

* Click on Downloads
* Click on windows host option here(under Virtualbox 7.0.8 platform packages heading) and install virtualBox.exe file
* Install Virtualbox on windows OS by clicking next buttons till Finish



# Click Files> Tools>Extensions Pack Manager. Then click ‘Install’ written on RHS to install Extension pack ,browse extension pack file downloaded earlier in downloads(or other location), double click or open it. It starts installation of Oracle VM Virtualbox extension pack 7.0 version downloaded earlier with virtualbox from same download page on website. Agree terms and Install it.

# Installation done successfully and we can go to File>Quit to quit or use it for work.

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

# Download ISO File of Ubuntu

For this-

1. Type ubuntu on google and search.Click the link for download. Click **Download ubuntu 24.04 LTS version**(Latest version now. LTS means Long Term Support version)

**Once iso file for ubuntu is downloaded** as ubuntu-22.04.1-desktop-amd64.iso then **its ready to be installed and used in virtualbox**.(It’s a big file more or less than around 3 GB, so take some long time to download)

# Installing Ubuntu in Virtualbox: Steps-

# Open Oracle VM Virtualbox, click new button on top( click 3 horizontal lines in fron of’tools’>Welcome>New

# Provide name of new machine and version say’ubuntu 20.04’ given here in my PC once the new button click opens the box.

# Below it is Folder path where all the ubuntu related files will be stored in your system.

# Below it lies Type which is Linux for ubuntu as ubuntu is a linux based OS.

# Below is Version as ubuntu64 as we have downloaded ubuntu 64 bit iso file.

# Click Next

# Preferably provide memory size upto the green mark using slidebar,then create next button

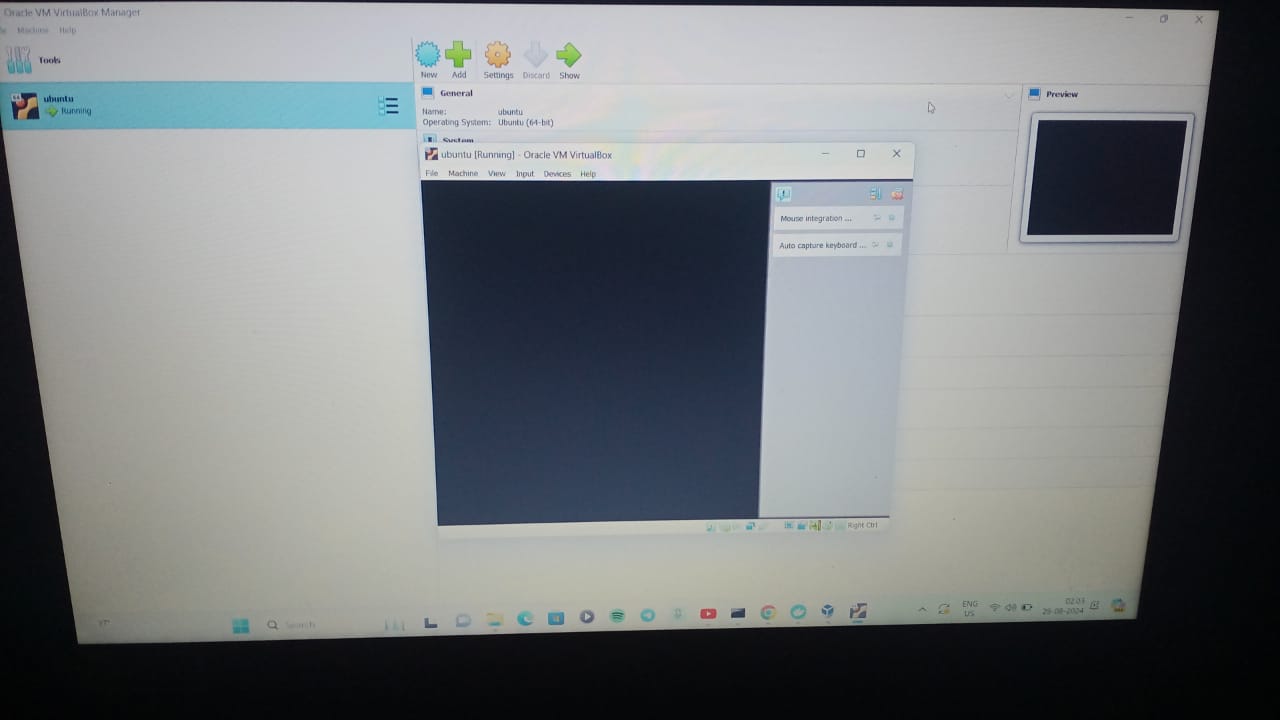
# Then leave everything as default and go on clicking next till finish.This shows on LHS of oraqcle VM virtualbox screen as installed ubuntu20.04 there on virtualbox.

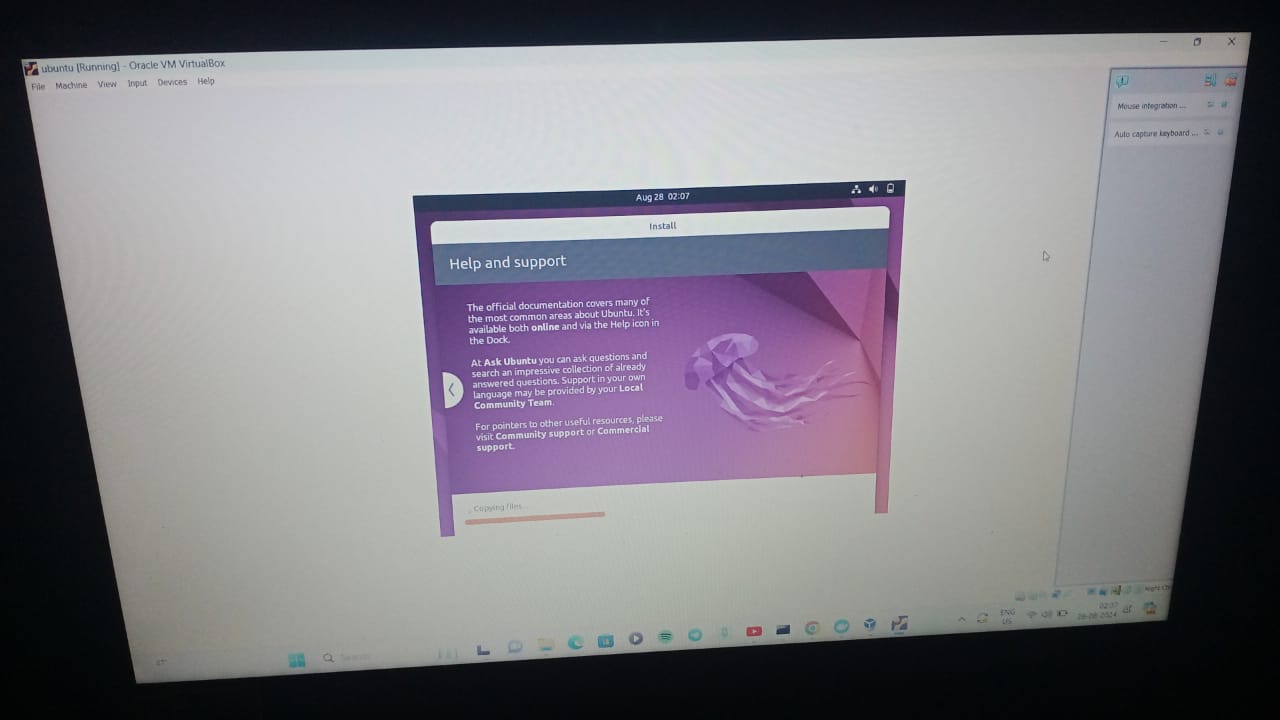
# Select by clicking the ubuntu machine created just and shown written on LHS of virtualbox ,don’t click start yet, first click settings on top RHS.Leave’ General ‘ as default if no change required.Goto ‘Advanced’ and select both ‘shared clipboard’ and ‘drag n drop’ options as ‘bidirectional’.This will allow copy-paste and other operations bidirectionally between our host OS windows 10 on my PC and ubuntu OS on virtualbox.

# Goto Storage>controller IDE - >Empty> click on Circular disk icon on RHS>choose a disk file> then Browse the location where ubuntu iso file is downloaded, copy the location and paste on virtualbox and double click or open ubuntu iso file to be uploaded in virtualbox.This shows ubuntu iso file uploaded in controller IDE> ubuntu iso(in place of ‘empty’ earlier).

# Do other settings if required or wish else leave them as default and click finally OK.Then ubuntu is ready for installation

# Then click on ubuntu in LHS, then click ‘start’ button on top RHS





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

To set up Visual Studio Code (VS Code) on an Ubuntu virtual machine (VM), Steps required:

* Download and install VMware Player
* Download an ISO image from Linux
* Create a new virtual machine
* Select the Ubuntu image and follow the steps
* Configure the VM
* Install VMware tools
* Install VS Code

We can also install VS Code on Ubuntu using the Ubuntu Software Center, a .deb package, or a Snap package:

* Ubuntu Software Center: Open the Ubuntu Software Center, find VS Code, click Install, and enter your administrator password
* .deb package: Download and install the 64-bit .deb package using the graphical software center or the command line
* Snap package: Install the Snap package, which automatically updates in the background

3.Set up Python. (5 marks)

* **Download the python version 3.10**, you may refer to below commands:

sudo apt update

sudo apt install software-properties-common

sudo add-apt-repository ppa:deadsnakes/ppa

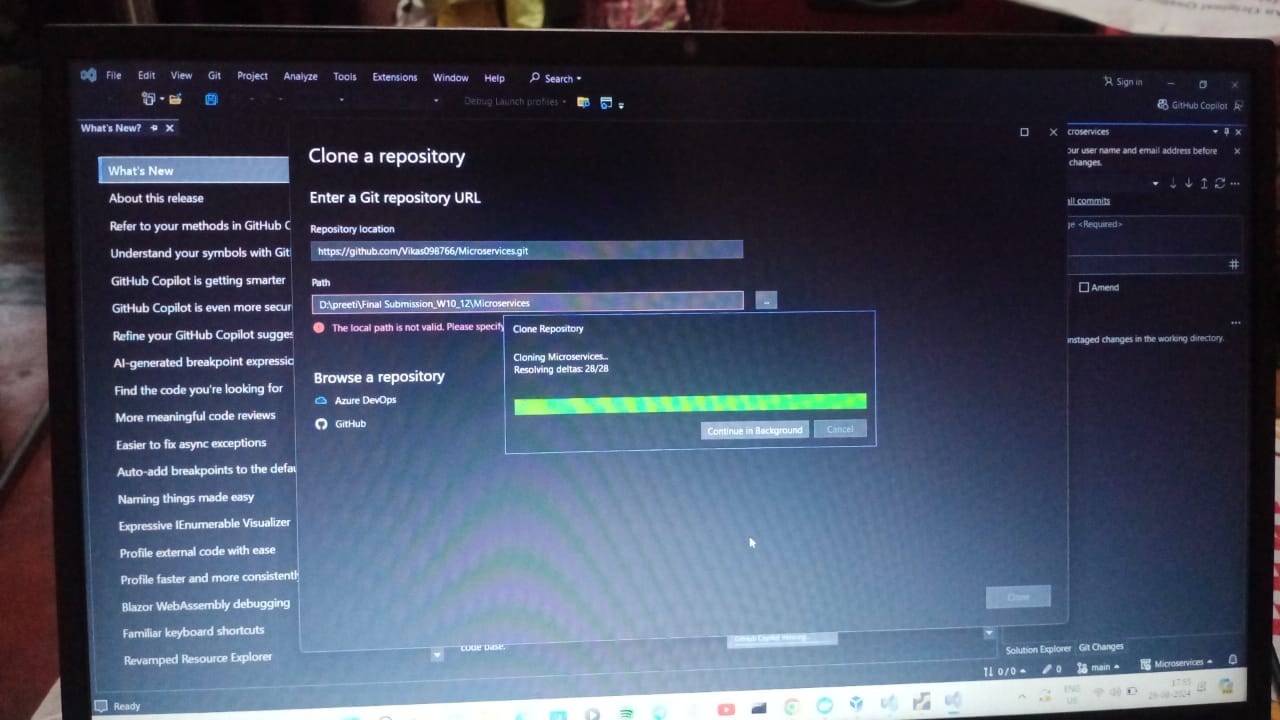
sudo apt install python 3.10

#Verification of the installation was successful.Python3 -Version

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

From Visual Studio Code clone the below Github repository:

<https://github.com/Vikas098766/Microservices.git>



Project Structure is below(Structure of the codes):

MICROSERVICES folder contains multiple folders as below-

* + 1. code\_model\_training

train.py (contains all codes related to training pipeline)

* data - breastcancer.csv
* model - model\_ binary.dat.gz (All the model weights are saved here)
* ms\_init\_.py
* functions.py (Related functions we keep here)
* tests - example\_calls.test (Any logins etc we will do here)
* app.py
* requirements.text (All required packages and corresponding versions are defined here)

5.Create a Virtual Environment. (1 mark)

* Once we **downloaded above github repository folder to our Ubuntu /Linux Machine** we have to do the following:
* **Virtual Environment :**

First we need **to create a virtual environment for the cloned github project**, to keep a track of every dependency, it is also useful to use an explicit version of Python.

* To setup a Python3 Virtual environment, navigate to your project folder on your terminal and type the following command:

python3 -m venv venv

This will **create a new virtual environment** named venv using the version of python3 that you have installed in your system.

* Next you need to **activate the virtual environment** by sourcing the activation script: command is-

Source venv/bin/activate

* After executing this command your prompt will change to indicate that you are now operating from within the virtual environment

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

* Now with the virtual environment **we can install the dependencies written in requirements .text**. Command is-

pip install –r requirements.text

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

* **Model training and saving the model**:

After we have installed all the dependencies we can now run the script in code\_model\_training/train.py , this script takes the input data and outputs a trained model and a pipeline for our web service. Command is-

Python code\_model\_training/train.py (We do the training here)

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

* Web application

Finally we can test our web application by running: Command is-

flask run –p 5000

Testing the application and making predictions:

The below command will run the development server locally and listen to port 5000

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

**Testing the application and making predictions**:

The below command will run the development server locally and listen to port 5000 where we can test our application! The folder ***/tests*** contain some example calls to test that our application is up and running:

**Example call**:

curl -X GET <http://localhost:5000/info>

**The service should respond**:

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

**Example call:**

POST method predict:

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, "concavepoints\_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, "concavepoints\_worst": 0.2654, "symmetry\_worst": 0.4601, "fractal\_dimension\_worst": 0.1189}]' \

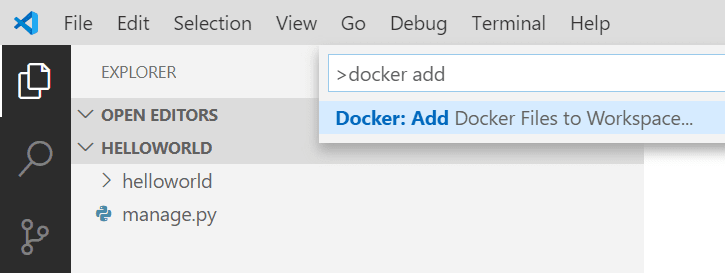
"Content-Type: application/json" \

-X POST <http://0.0.0.0:5000/predict>

**The service should respond:**

{“label”:”M”,”prediction”:1,”status”:200}

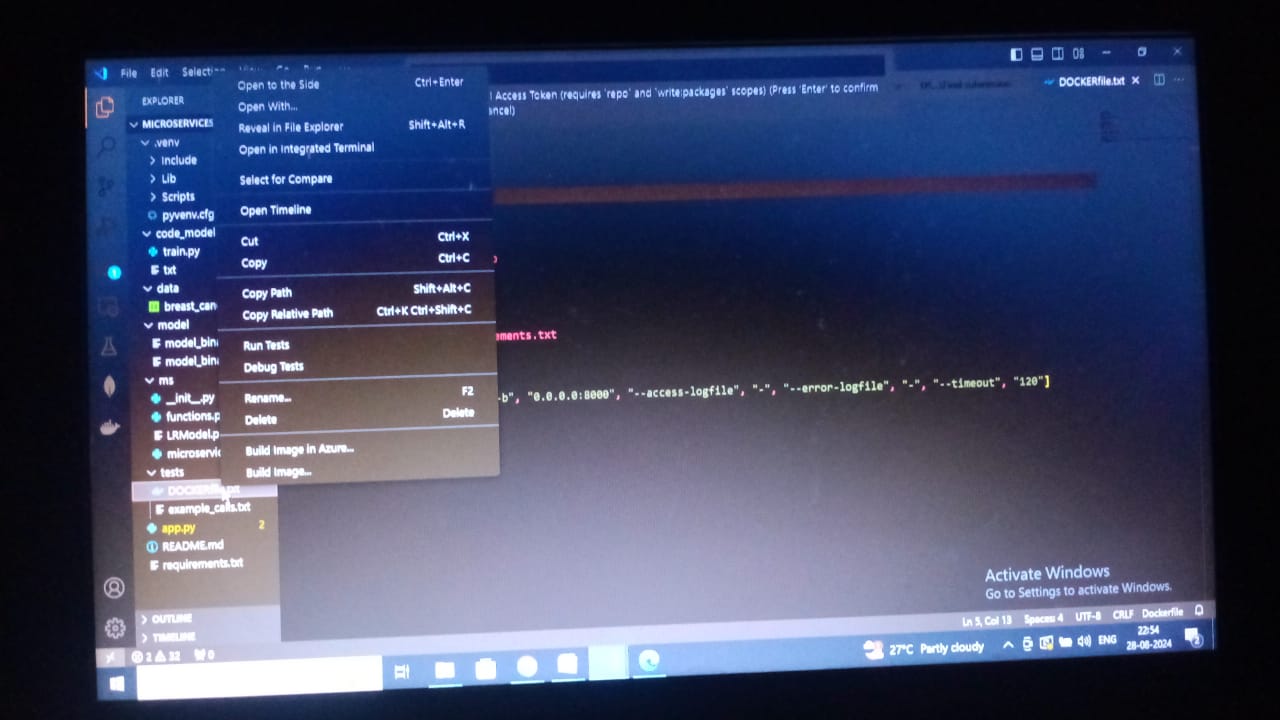
10.Create a docker image containing everything needed to run the application.(10 marks)

* [Install Docker](https://docs.docker.com/install/) on your machine and add it to the system path.
* On Linux, you should also [enable Docker CLI for the non-root user account](https://docs.docker.com/install/linux/linux-postinstall/#manage-docker-as-a-non-root-user) that will be used to run VS Code.
* The Docker extension. To install the extension, open the Extensions view (Ctrl+Shift+X), search for docker to filter results and select Docker extension authored by Microsoft.
* Open the project folder in VS Code.
* Open the **Command Palette** (Ctrl+Shift+P) and choose **Docker: Add Docker Files to Workspace...**:
* 
* When prompted for the app type, select **Python: Django**, **Python: Flask**, or **Python: General** as the app type. For this tutorial, we'll focus on the **Python: General** case, but will also include notes for Django and Flask.
* Enter the relative path to the app's entry point. This excludes the workspace folder you start from. If you created a python app with inference.py or train.py , choose that.
* **Flask**: Choose the path to where you create your Flask instance.Or You may also enter the path to a folder name as long as this folder includes a \_\_main\_\_.py file.
* Select the port number.. Any unused will port(preferably above 1024 to avoid root user concerns), but Django and Flask use standard default ports.

**Django**: The default port 8000.

**Flask**: The default port is 5000.

* **[Important Note**: To use our setup, the Python framework (Django/Flask) and Gunicorn **must be included** in the requirements.txt file. If the virtual environment/host machine already has these prerequisites installed and is supposed to be identical to the container environment, ensure app dependencies are ported over by running pip freeze > requirements.txt in the terminal. **This will overwrite your current requirements.txt file.]**
* **To Create image: Right click the docker file in our project and select build image**



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

The below command will run the development server locally and listen to port 5000 where we can test our application! The folder ***/tests*** contain some example calls to test that our application is up and running:

**Example call**:

curl -X GET <http://localhost:5000/info>

**The service should respond**:

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

**Example call:**

POST method predict:

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, "concavepoints\_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, "concavepoints\_worst": 0.2654, "symmetry\_worst": 0.4601, "fractal\_dimension\_worst": 0.1189}]' \

"Content-Type: application/json" \

-X POST <http://0.0.0.0:5000/predict>

**The service should respond:**

{“label”:”M”,”prediction”:1,”status”:200}

