Name: Fithi Ghebreamlak

ID: 20068

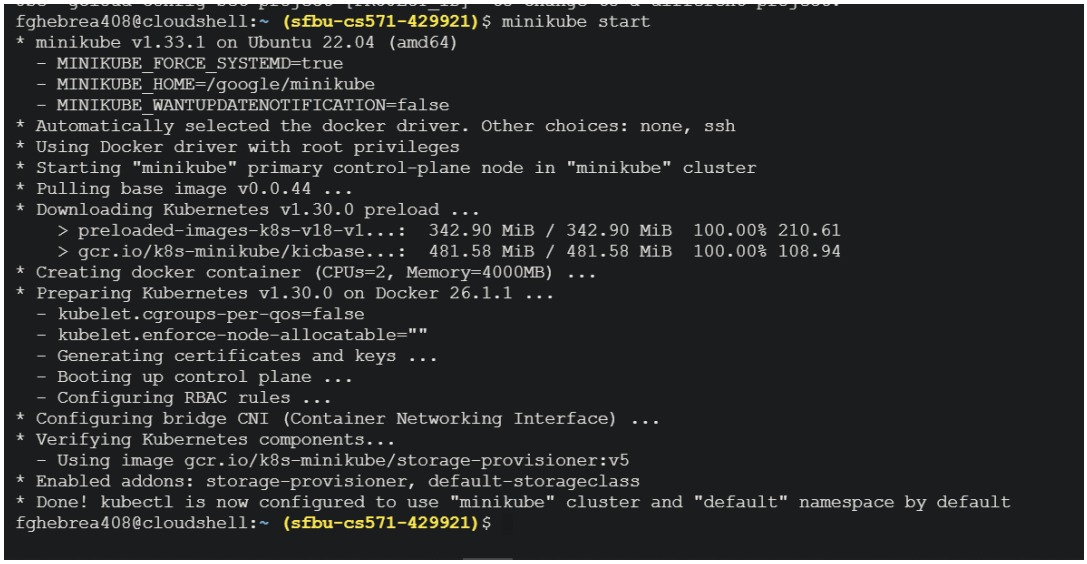
CS501- Week 10 Homework 1: Machine Learning on Kubernetes

https://hc.labnet.sfbu.edu/~henry/sfbu/course/cloud_computing/genai/slide/exercise_kubernetes.html

Machine Learning on Kubernetes

Creating and uploading necessary files in GCP- Cloud Shell Terminal

1. Start minikube in Google Cloud Platform



2. Create requirements.txt file using the following command

- sudo vim requirements.txt

Then enter the following contents

Flask==1.1.1

gunicorn==19.9.0

itsdangerous==1.1.0

Jinja2==2.10.1

MarkupSafe==1.1.1

Werkzeug==0.15.5

numpy==1.19.5 # Adjusted to a version before np.float deprecation

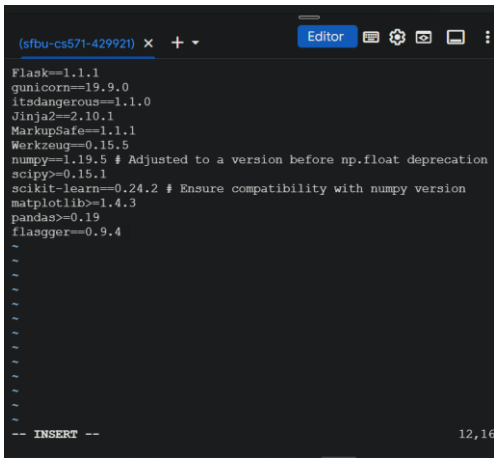
scipy>=0.15.1

scikit-learn==0.24.2 # Ensure compatibility with numpy version

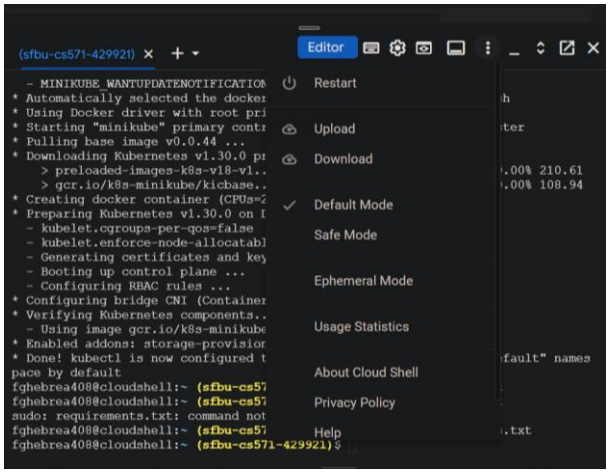
matplotlib>=1.4.3

pandas>=0.19

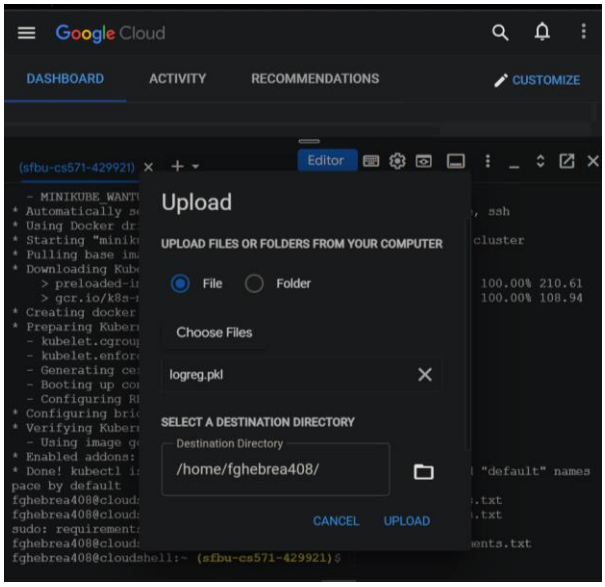
flasgger==0.9.4



3. Upload logreg.pkl file by clicking the three dots in the top-right part of the Cloud Shell Terminal and then choose upload



Then upload the logreg.pkl file as following



4. Create flask_api.py file using the command

- sudo vim flask_api.py

Then enter the following contents

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 25 12:50:04 2020
@author: pramod.singh
"""
from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
from flasgger import Swagger
app = Flask(__name__)
Swagger(app)
pickle_in = open("logreg.pkl", "rb")
model = pickle.load(pickle_in)
@app.route('/')
def home():
 return "Welcome to the Flask API!"
```

```
@app.route('/predict', methods=["GET"])
def predict_class():
 """Predict if Customer would buy the product or not.
 ---
 parameters:
 - name: age
 in: query
 type: number
 required: true
 - name: new_user
 in: query
 type: number
 required: true
 - name: total_pages_visited
 in: query
 type: number
 required: true
 responses:
 200:
 description: Prediction
 """
 age = int(request.args.get("age"))
 new_user = int(request.args.get("new_user"))
 total_pages_visited = int(request.args.get("total_pages_visited"))
 prediction = model.predict([[age, new_user, total_pages_visited]])
 return "Model prediction is " + str(prediction)
@app.route('/predict_file', methods=["POST"])
def prediction_test_file():
 """Prediction on multiple input test file.
```

---

parameters:

- name: file

in: formData

type: file

required: true

responses:

200:

description: Test file Prediction

"""

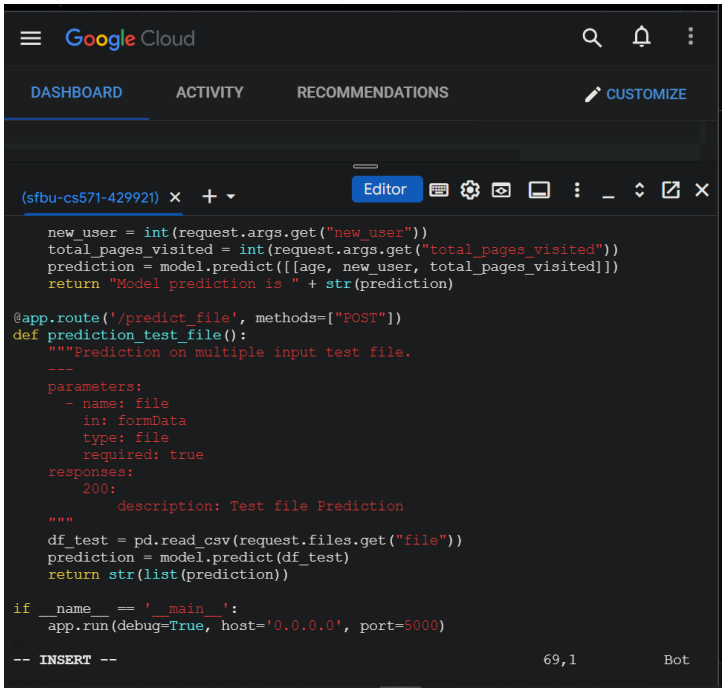
df_test = pd.read_csv(request.files.get("file"))

prediction = model.predict(df_test)

return str(list(prediction))

if __name__ == '__main__':

app.run(debug=True, host='0.0.0.0', port=5000)



Step 4: Dockerfile

1. Create Dockerfile using command

- sudo vimi Dockerfile

Then enter the following content

FROM python:3.8-slim

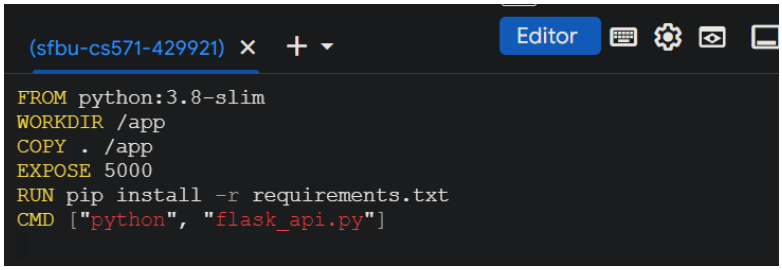
WORKDIR /app

COPY . /app

EXPOSE 5000

RUN pip install -r requirements.txt

CMD ["python", "flask_api.py"]



1. 'FROM python:3.8-slim'

• This line sets the base image for the Docker image you are creating. It tells Docker to start with the 'python:3.8-

slim' image, which is an official Python image with Python 3.8 installed on it. The 'slim' version is a smaller version

of the image that has fewer packages pre-installed, making the image size smaller.

2. 'WORKDIR /app'

• This instruction sets the working directory within the Docker container to */app*. All subsequent commands will

be executed in this directory within the container.

3. 'COPY . /app'

• This line copies everything from the current directory (on the host machine where you're running the Docker build

command, indicated by the first**) into the */app directory inside the Docker image (the second ' /app*).

4. 'EXPOSE 5000'

• The 'EXPOSE' instruction informs Docker that the container listens on the specified network port at runtime. In

this case, it tells Docker that the container will listen on port 5000. It's worth noting that this does not actually

publish the port—it serves as documentation and is used by the 'docker run -p' command to map the container port to

a port on the Docker host.

5. 'RUN pip install -r requirements.txt*

• This command tells Docker to run pip install' inside the container, which will install the Python dependencies

listed in the requirements.tt file. These dependencies are necessary for the Flask application to run correctly.

6. CMD ["python", "flask_api.py"]'

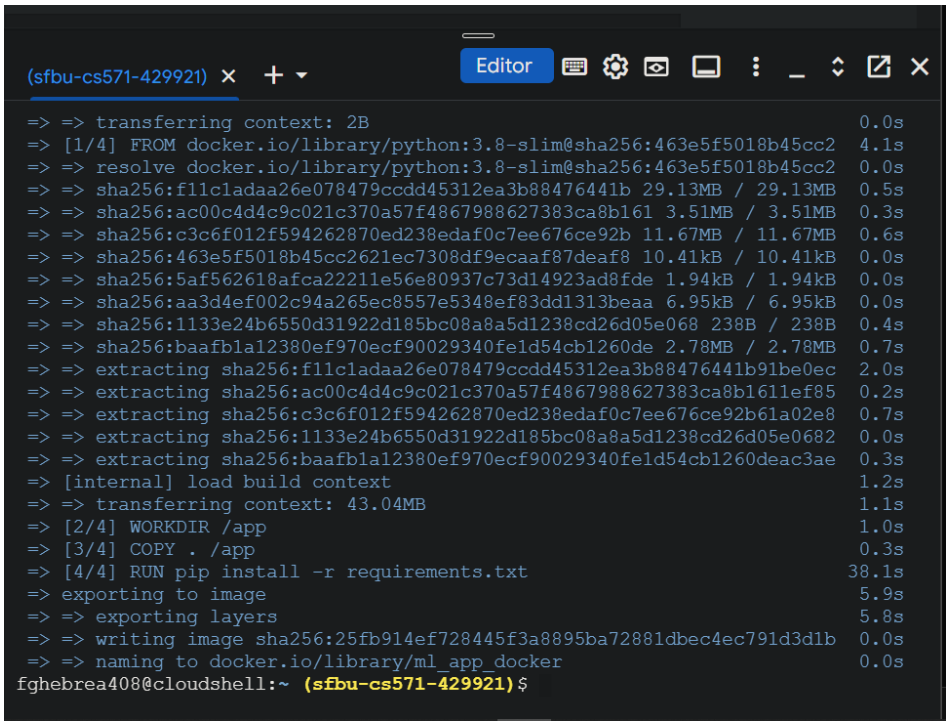
• This is the command that will be executed by default when the Docker container starts. In this case, it's telling

Docker to run 'flask_api.py using Python. This is the Flask application you want to run inside the container.

Step 5: Running the Docker Container

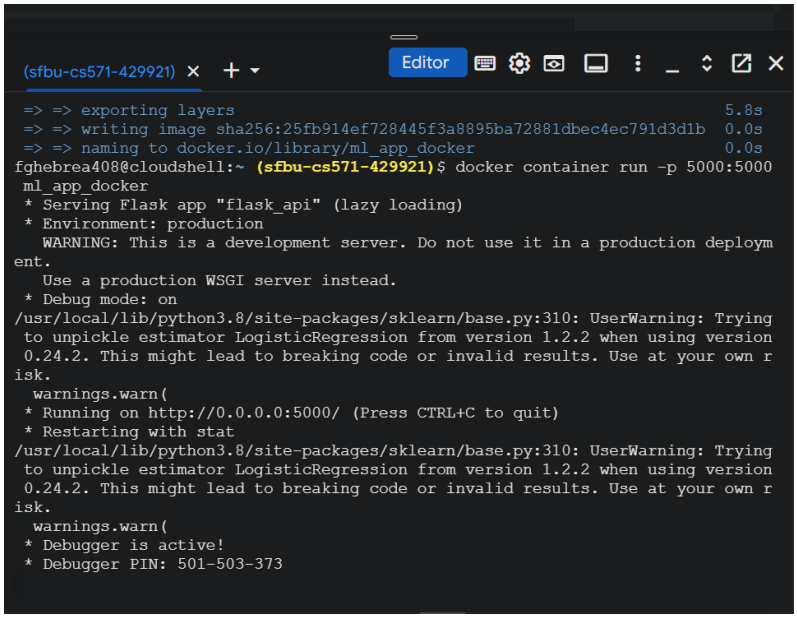
1. To build the docker image use the command

- sudo docker build -t ml_app_docker .

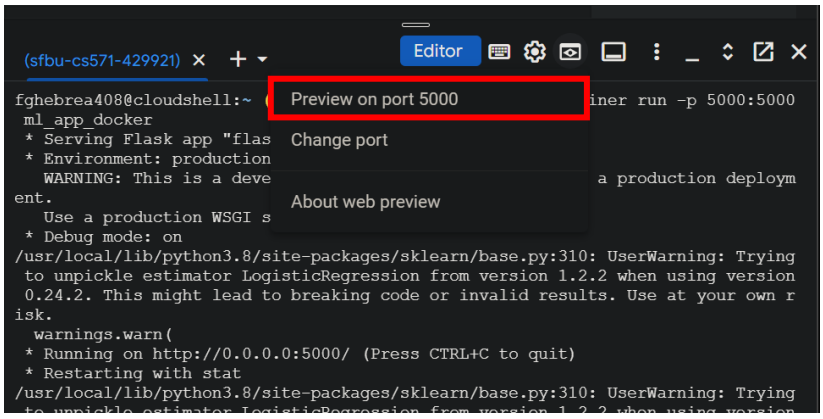


2. This command runs a Docker container from the ml_app_docker image:

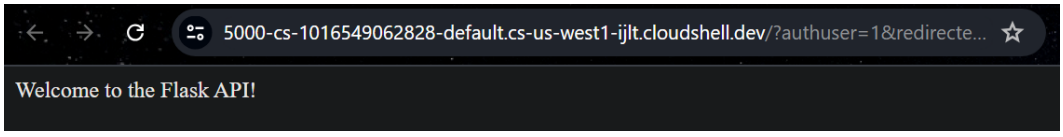
- docker container run -p 5000:5000 ml_app_docker



3. In the right-upper side of the terminal click the eye shaped button and then click Preview on port 5000. Change port if it is not 5000 by default.

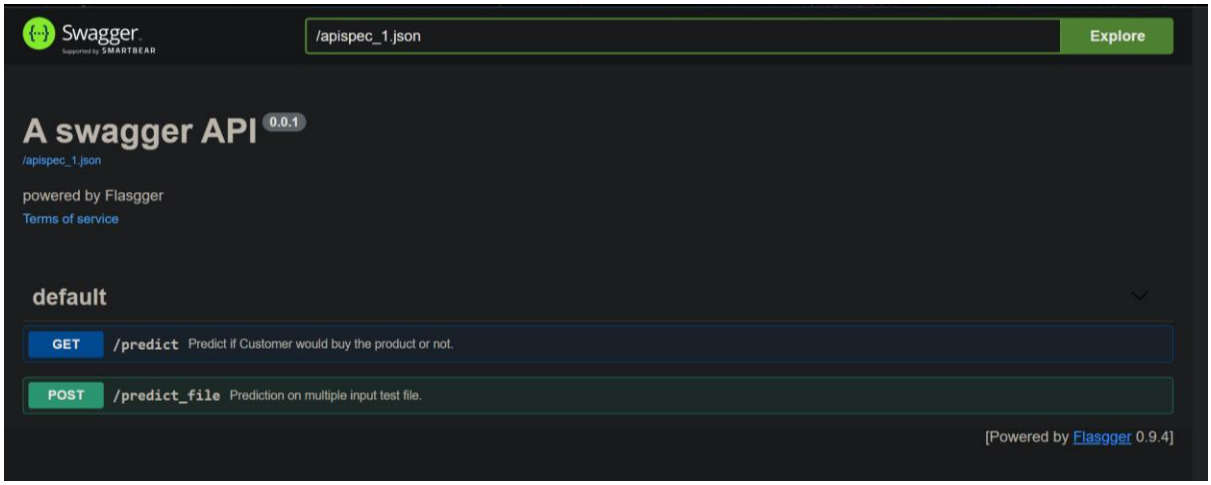


4. You will see this using the web preview.

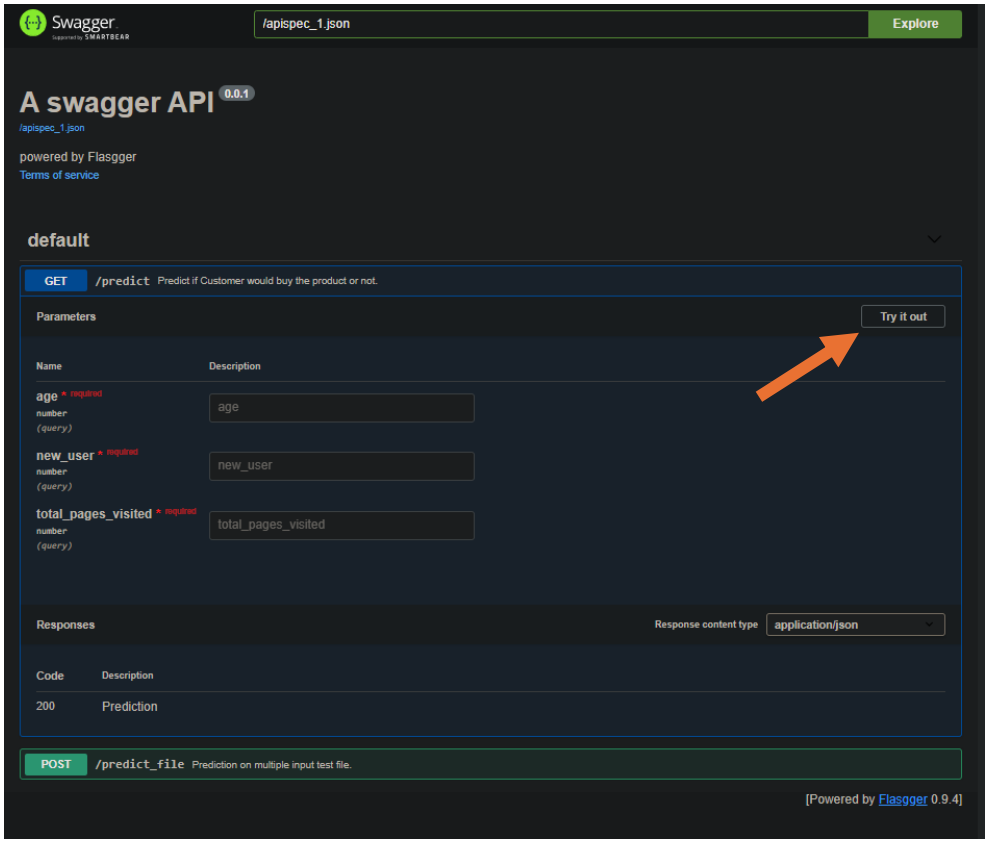


5. Add /apidocs/ at the end of the link to access the running ml- app as following

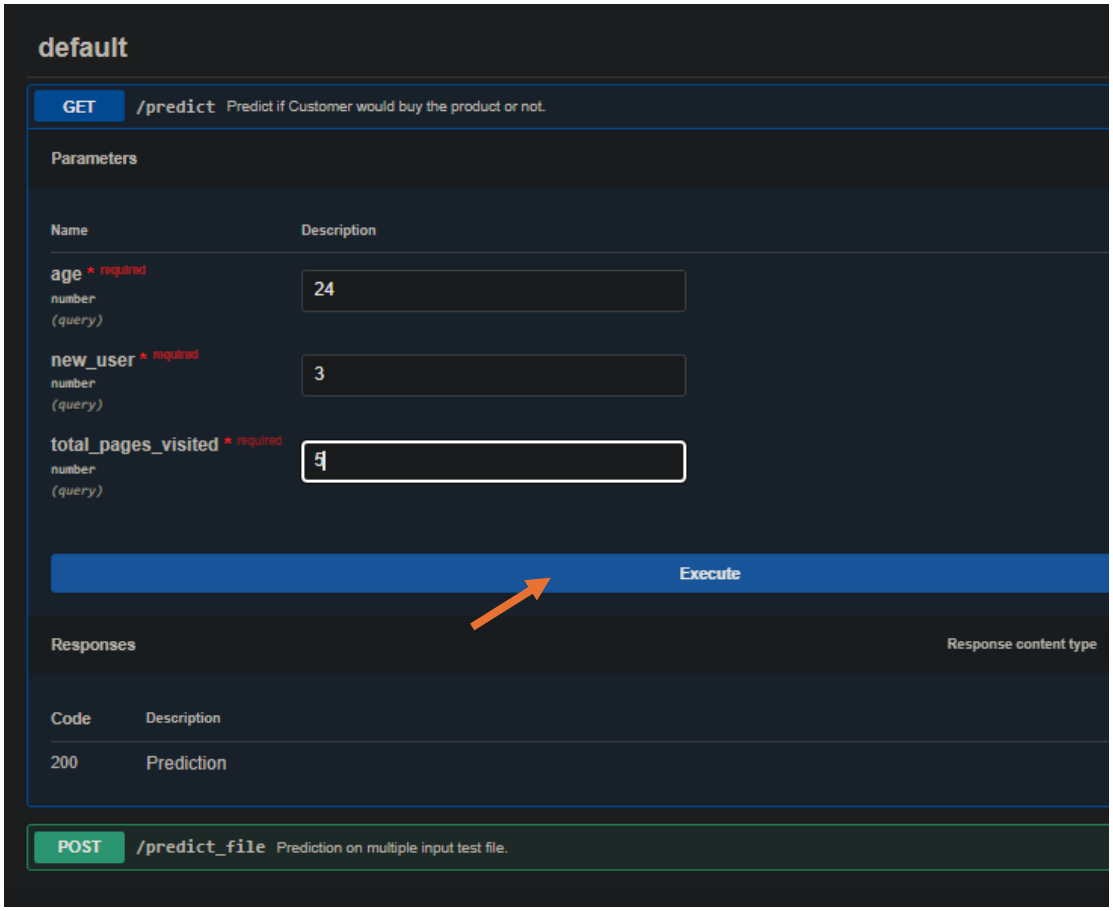
- There are two tabs GET and POST.



6. Click GET and then click Try it out in the top-right corner of the GET box.

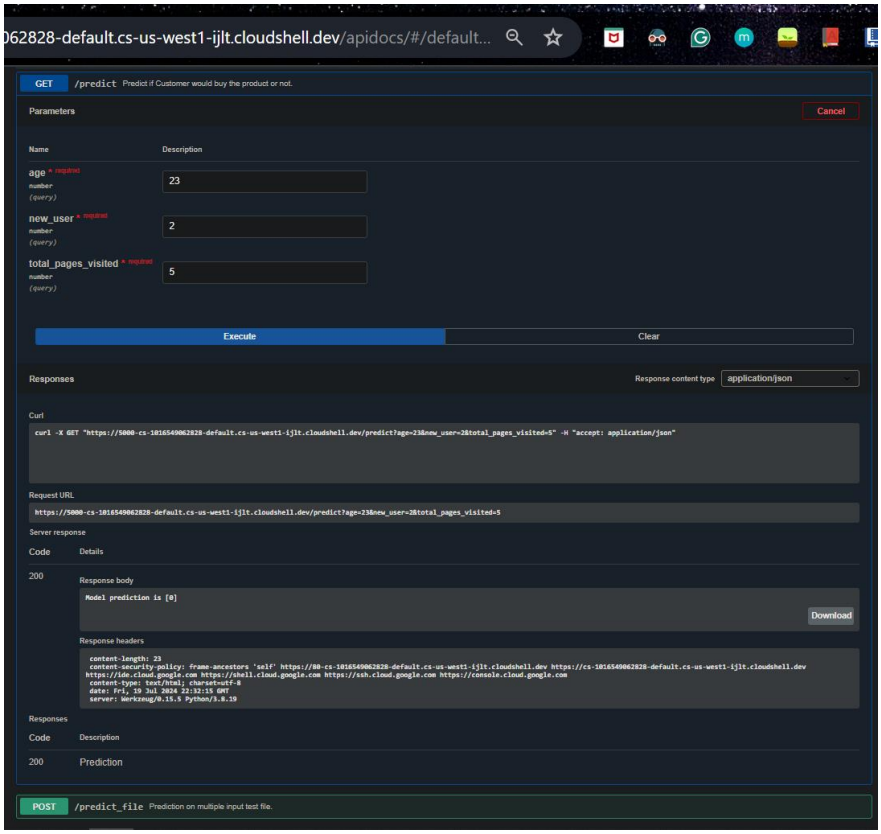


7. Fill values for the input parameters and then click Execute.

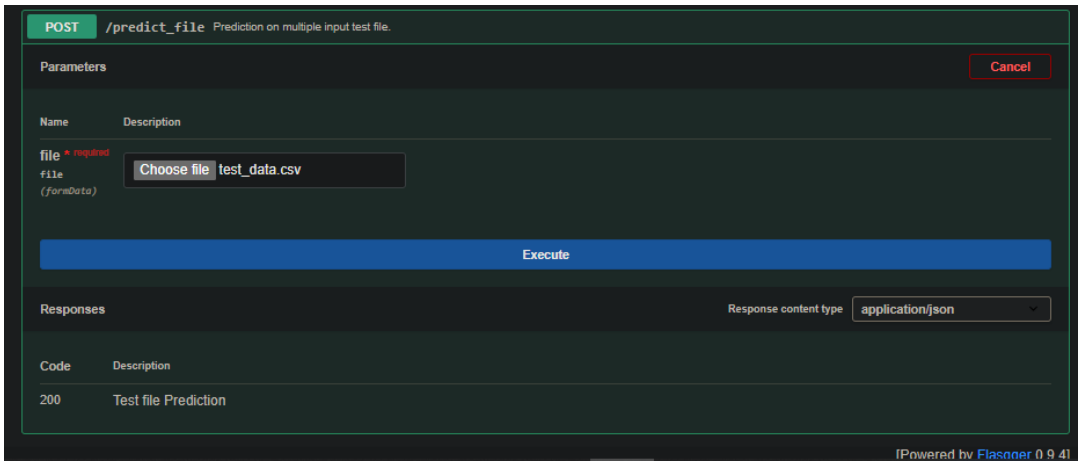


8. Upon the execution call, the request goes to the app, and predictions are made by the model.

- The result of the model prediction is displayed in the Prediction section of the page as following

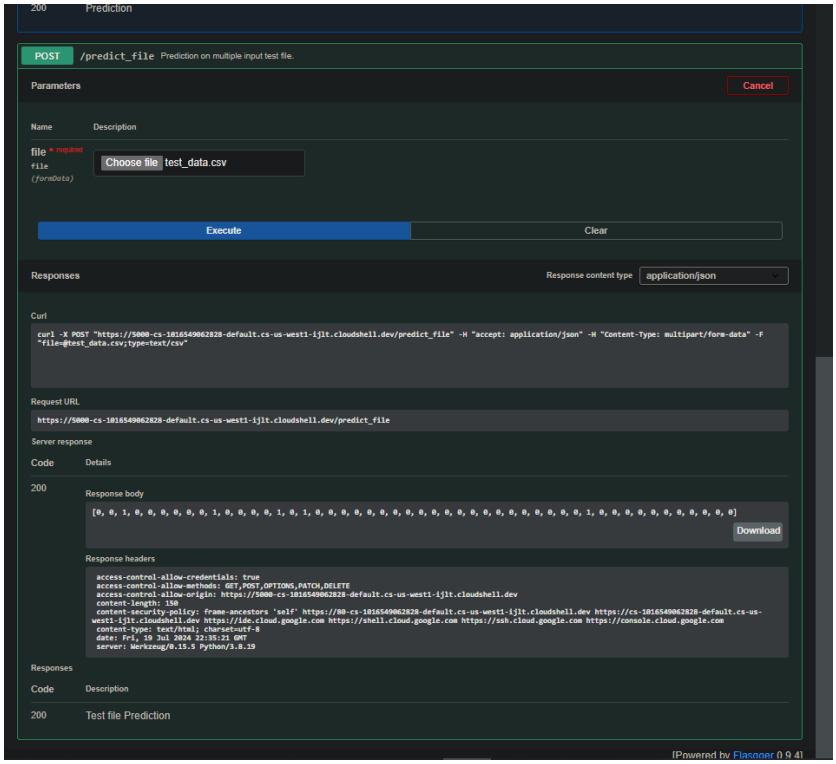


9. The next prediction that can be done is for a group of customers (test data) via a post request.



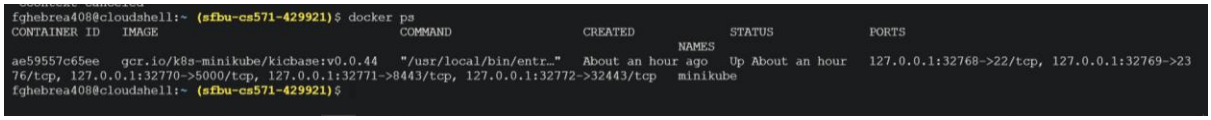
10. Upload the test data file containing the same parameters in a similar order.

The model would make the prediction, and the results would be displayed upon execute as following.



Step 6: Stopping/killing the running container

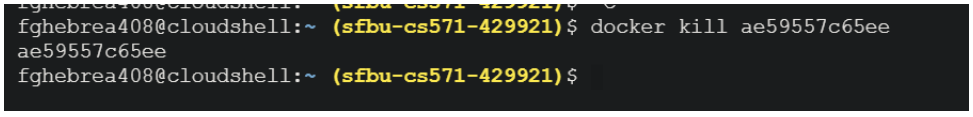
1. Use docker ps to list running Docker containers



- The CONTAINER_ID is given as 367119b87a37

2. Use the command

- docker kill <CONTAINER ID> to kill the running container as follows.



Updating Portfolio- GitHub link