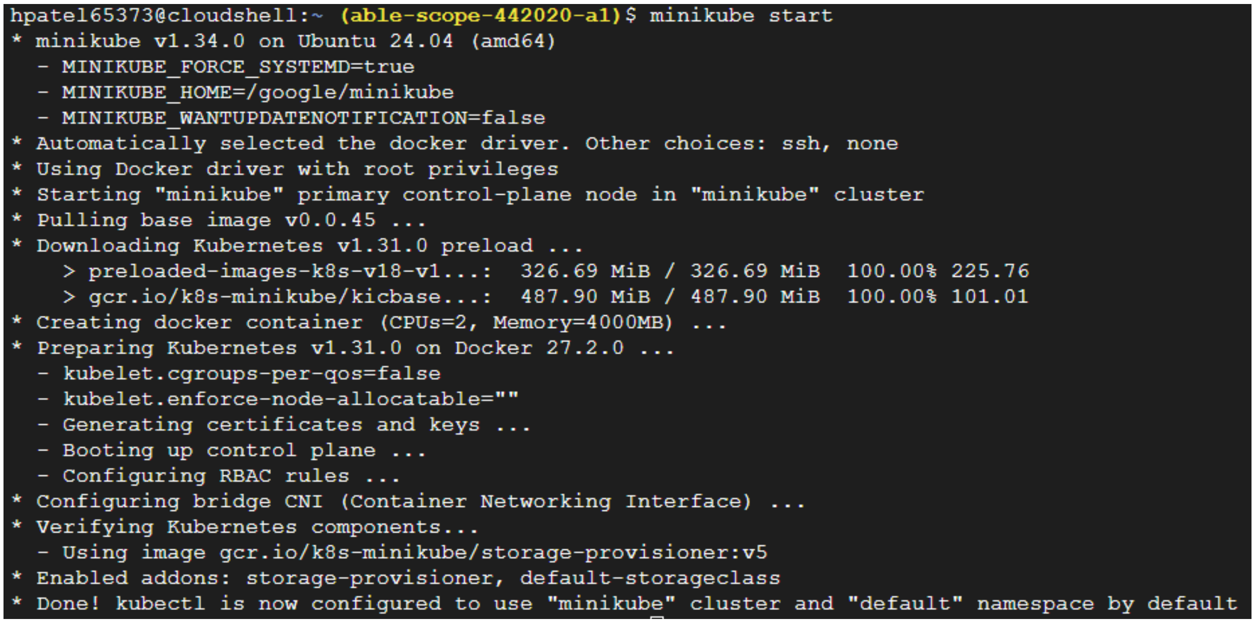
Machine Learning on Kubernetes

Creating and uploading necessary files in GCP- Cloud Shell Terminal

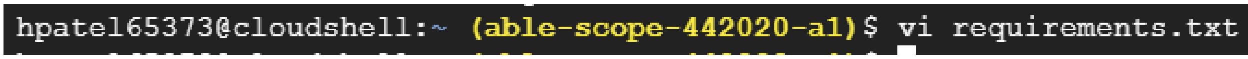
1 Start minikube in Google Cloud Platform



```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ minikube start
* minikube v1.34.0 on Ubuntu 24.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: ssh, none
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Downloading Kubernetes v1.31.0 preload ...
    > preloaded-images-k8s-v18-v1...:  326.69 MiB / 326.69 MiB  100.00% 225.76
    > gcr.io/k8s-minikube/kicbase...:  487.90 MiB / 487.90 MiB  100.00% 101.01
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

2 Create requirements.txt file using the following command

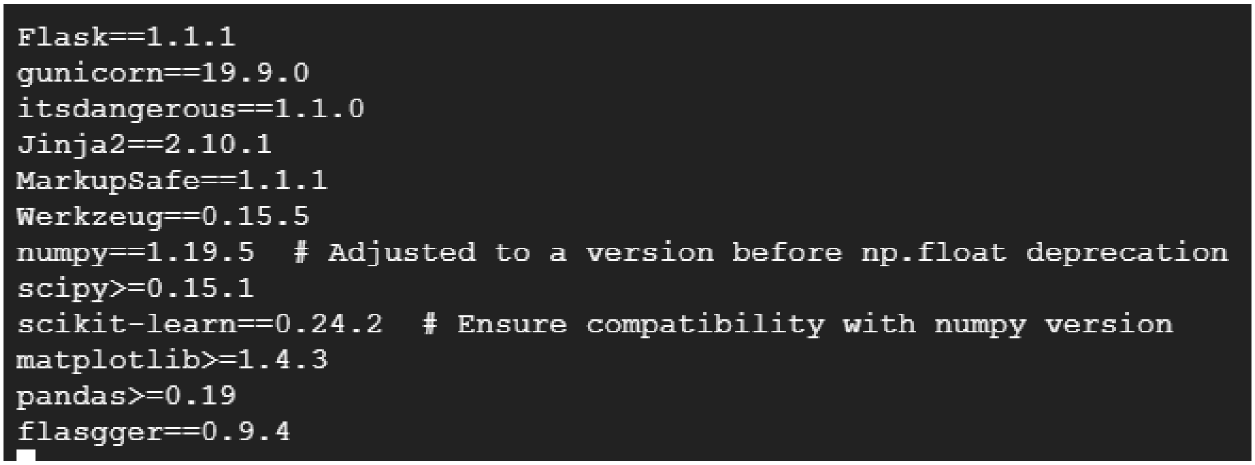
- vi requirements.txt



```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ vi requirements.txt
```

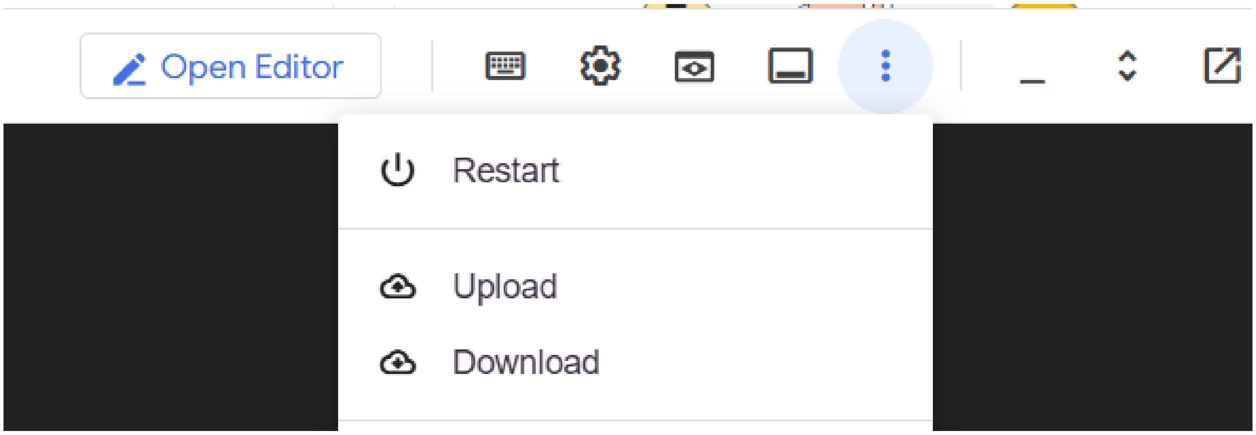
Then enter the following contents

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

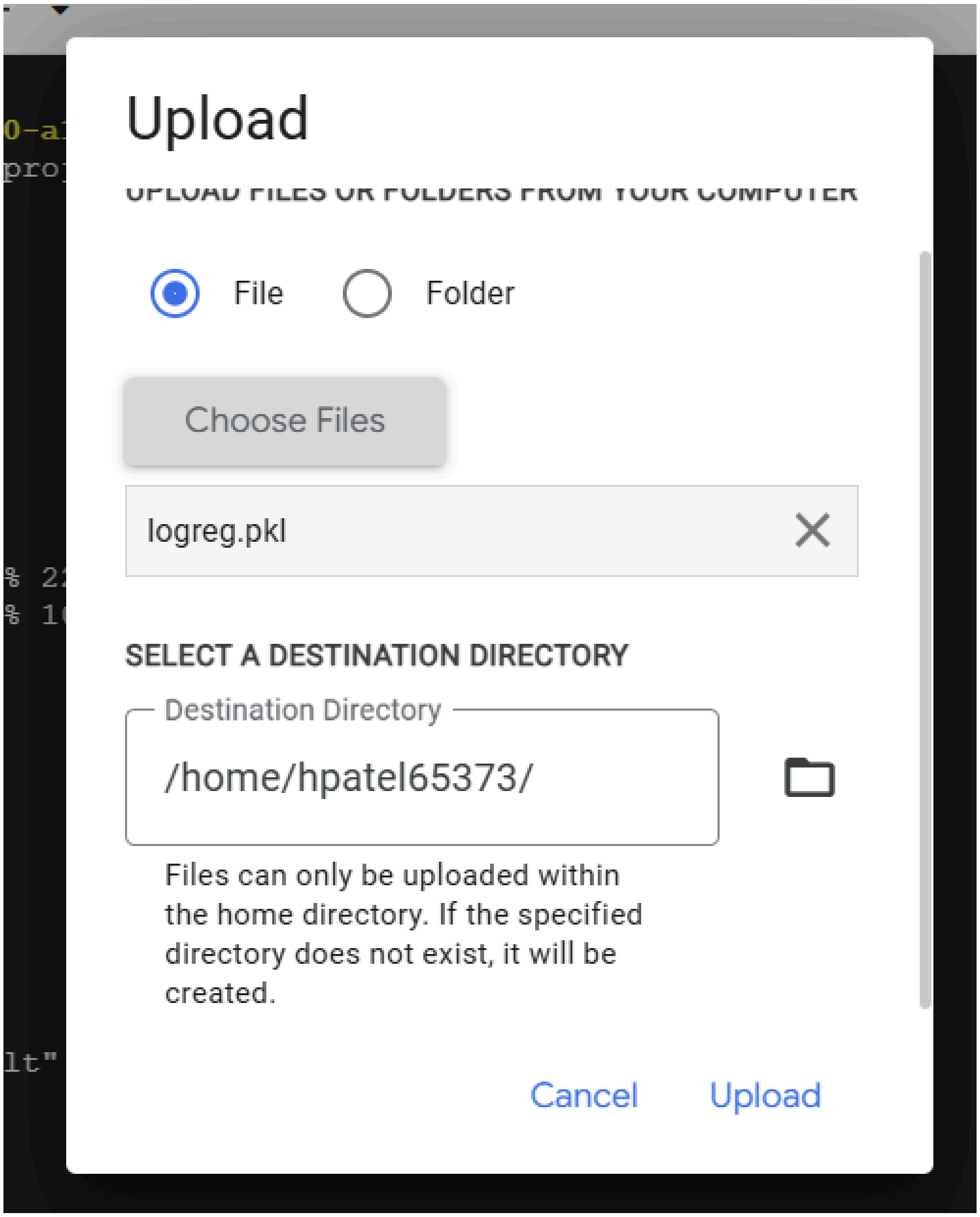


```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy==1.19.5  # Adjusted to a version before np.float deprecation
scipy>=0.15.1
scikit-learn==0.24.2  # Ensure compatibility with numpy version
matplotlib>=1.4.3
pandas>=0.19
flasgger==0.9.4
```

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



Upload

UPLOAD FILES OR FOLDERS FROM YOUR COMPUTER

● File ○ Folder

logreg.pkl

SELECT A DESTINATION DIRECTORY

Destination Directory: /home/hpatel65373/

Files can only be uploaded within the home directory. If the specified directory does not exist, it will be created.

4 Create flask_api.py file using the command

- vi flask_api.py

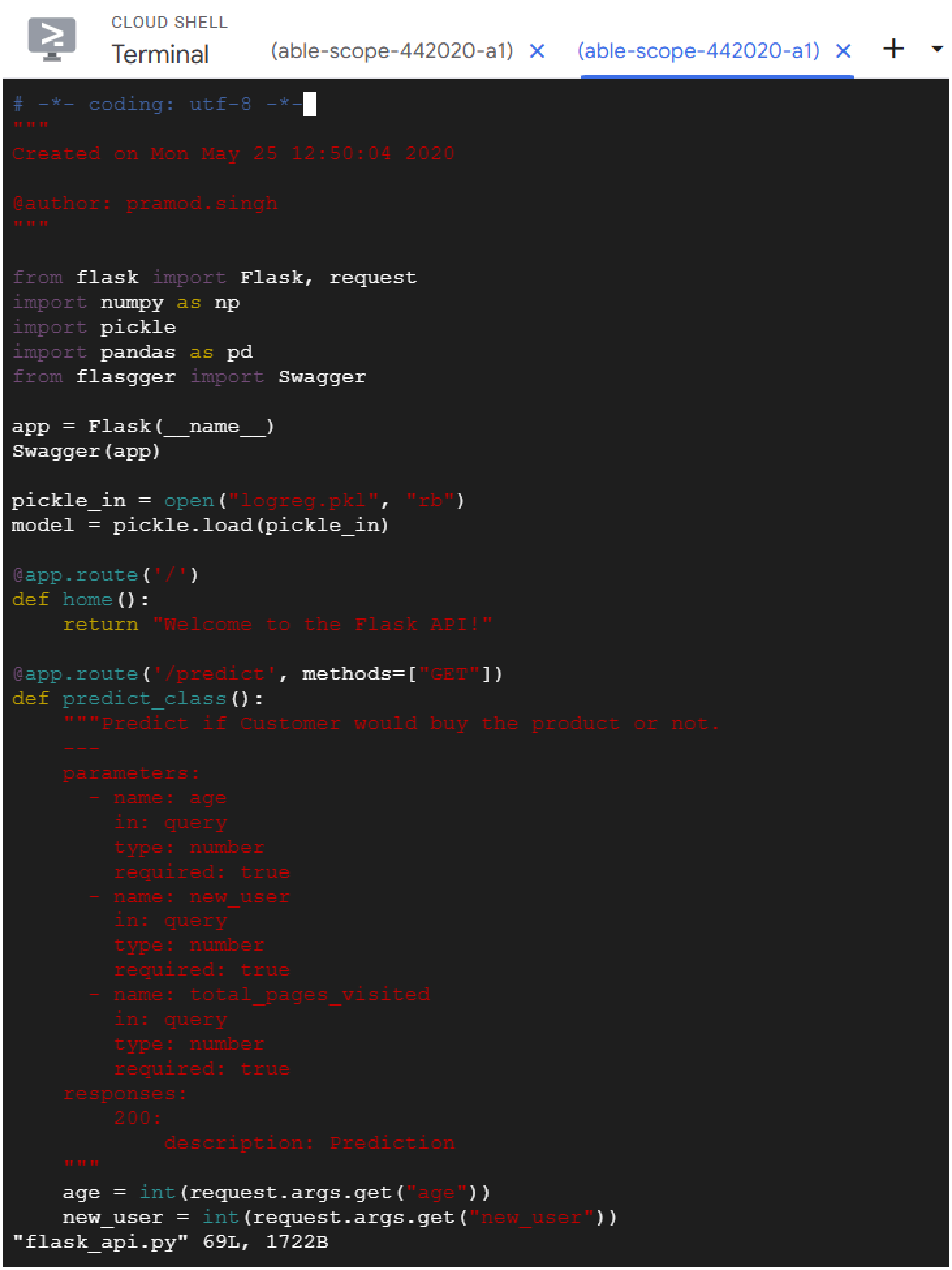
```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ vi flask_api.py
```

Then enter the following contents
# -*- coding: utf-8 -*-
"""
Created on Mon May 25 12:50:04 2020

@author: pramod.singh

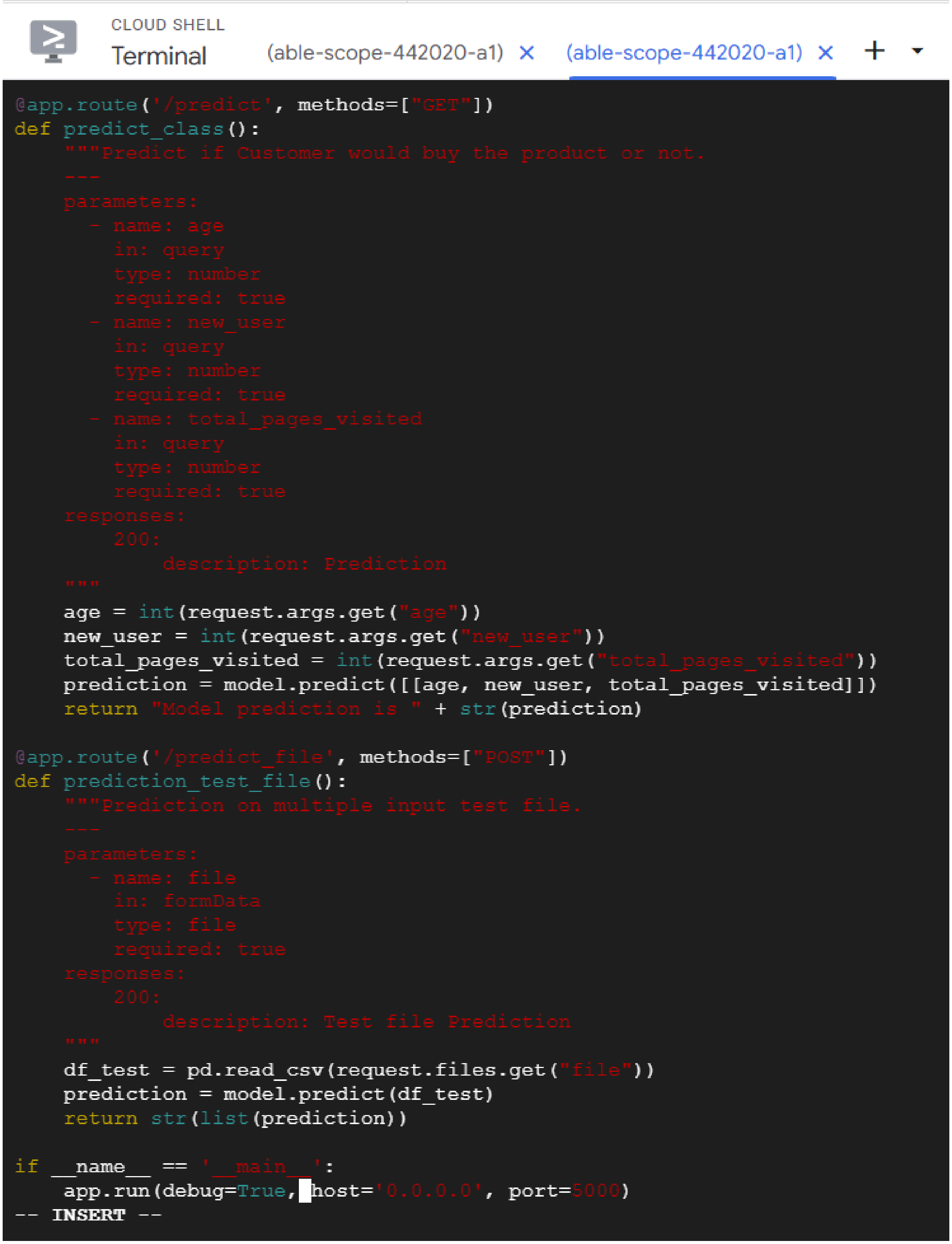
```
"""

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)
```

Machine Learning on Kubernetes



```
# -*- 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"))
"flask_api.py" 69L, 1722B
```

Machine Learning on Kubernetes

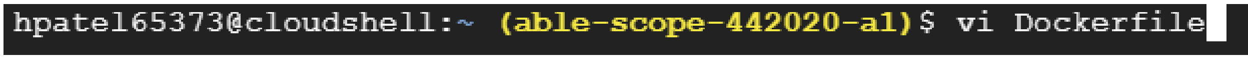


```
@app.route('/predict', methods=["GET"])
def predict_class():
    """Predict if Customer would buy the product or not.
    ---
    parameters:
      - name: age
        in: query
        type: number
        required: true
      - name: new_user
        in: query
        type: number
        required: 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)
-- INSERT --
```

Machine Learning on Kubernetes

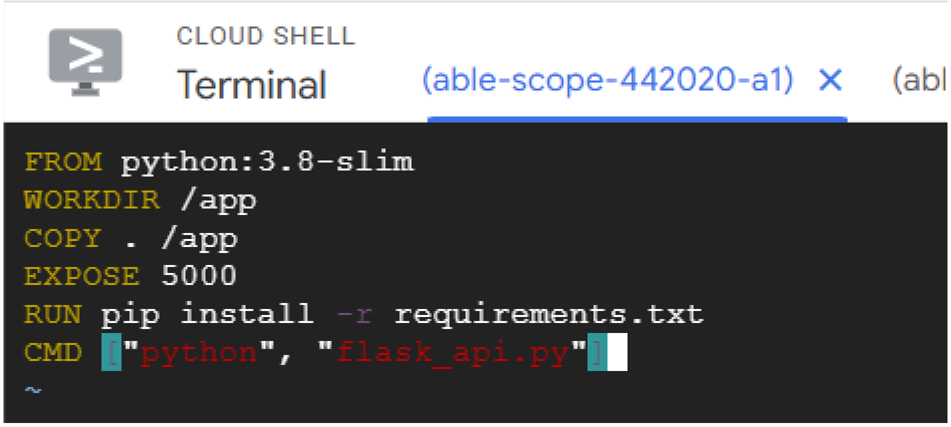
Step 4 Dockerfile

1. Create Dockerfile using command
   - vi Dockerfile



```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ vi 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"]



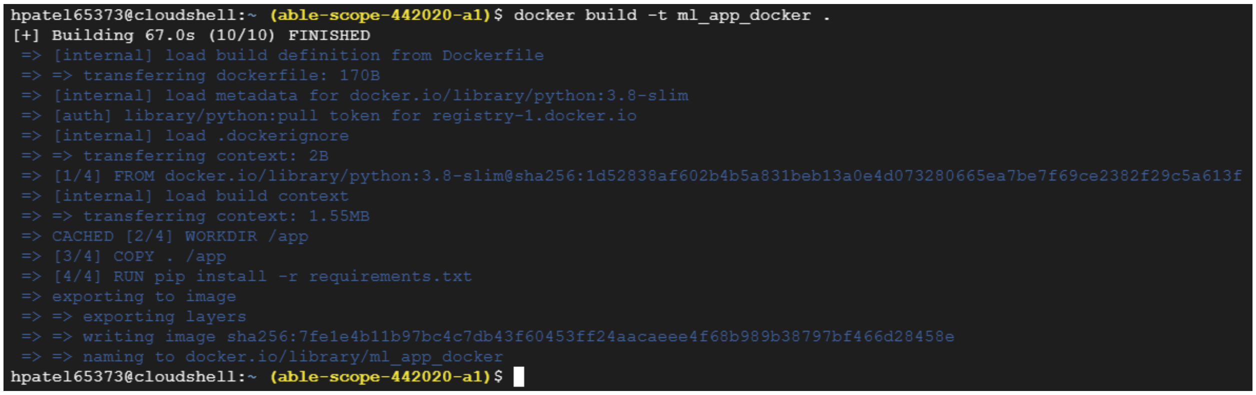
```
FROM python:3.8-slim
WORKDIR /app
COPY . /app
EXPOSE 5000
RUN pip install -r requirements.txt
CMD ["python", "flask_api.py"]
~
```

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.

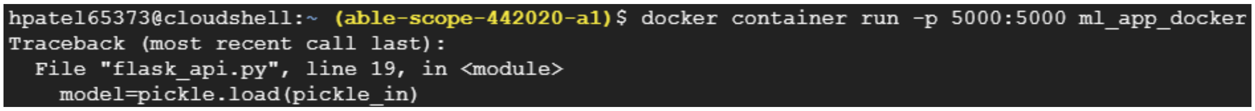
Step 5: Running the Docker Container

1. To build the docker image use the command
    - docker build -t ml_app_docker .



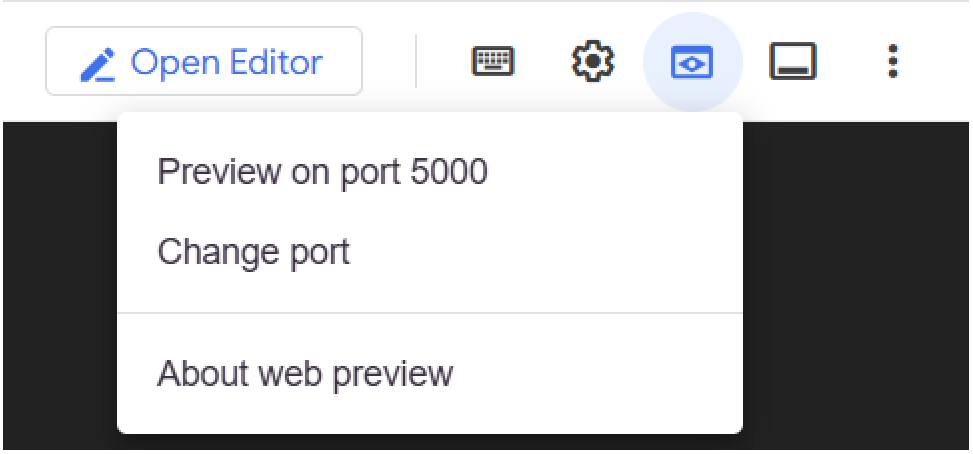
```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker build -t ml_app_docker .
[+] Building 67.0s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 170B
 => [internal] load metadata for docker.io/library/python:3.8-slim
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f
 => [internal] load build context
 => => transferring context: 1.55MB
 => CACHED [2/4] WORKDIR /app
 => [3/4] COPY . /app
 => [4/4] RUN pip install -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:7fe1e4b11b97bc4c7db43f60453ff24aacaeee4f68b989b38797bf466d28458e
 => => naming to docker.io/library/ml_app_docker
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

2. This command runs a Docker container from the ml_app_docker image:
    - docker container run -p 5000:5000 ml_app_docker

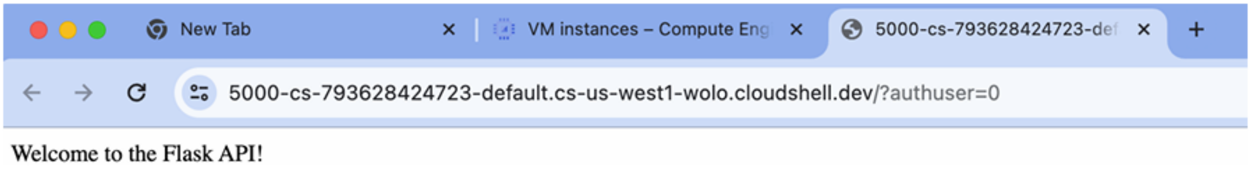


```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker container run -p 5000:5000 ml_app_docker
Traceback (most recent call last):
  File "flask_api.py", line 19, in <module>
    model=pickle.load(pickle_in)
```

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.



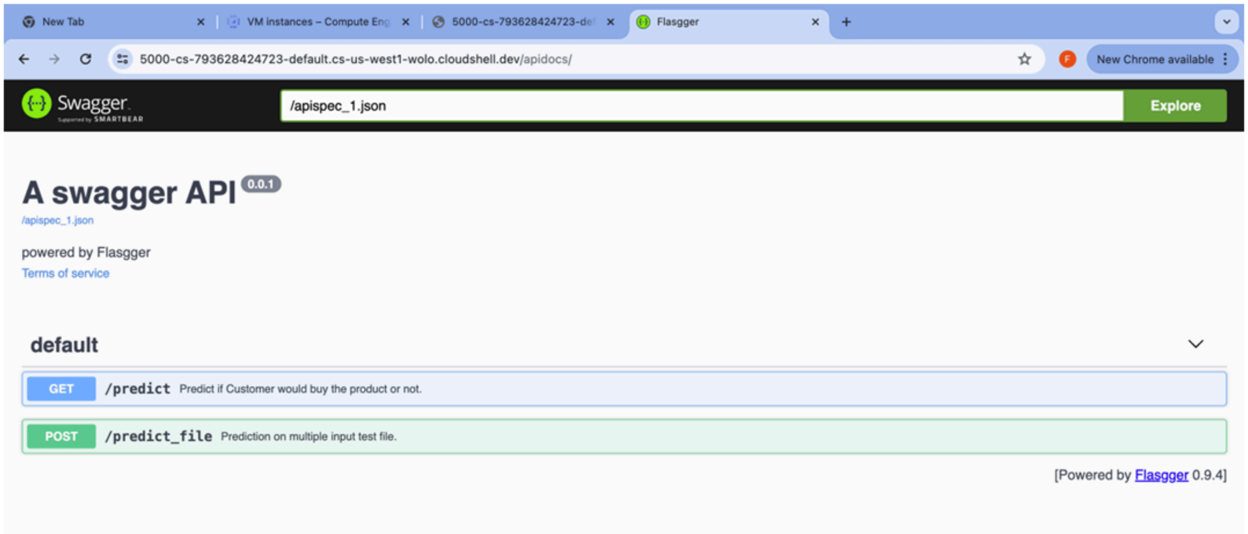
4. You will see this using the web preview.



5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/?authuser=0

Welcome to the Flask API!

5. Add /apidocs/ at the end of the link to access the running ml- app as following
    - Two tabs GET and POST



5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/apidocs/

/apispec_1.json

A swagger API 0.0.1

/apispec_1.json

powered by Flasgger

Terms of service

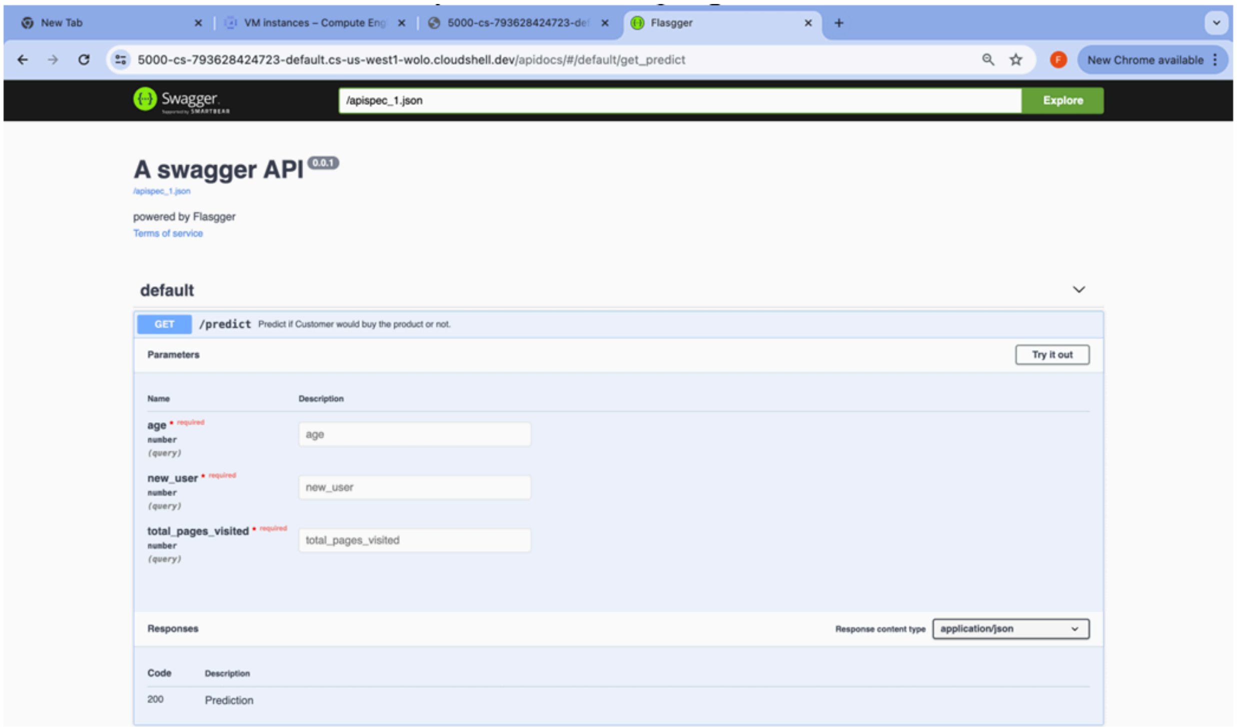
default

GET /predict Predict if Customer would buy the product or not.

POST /predict_file Prediction on multiple input test file.

[Powered by Flasgger 0.9.4]

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



5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/apidocs/#/default/get_predict

/apispec_1.json

A swagger API 0.0.1

/apispec_1.json

powered by Flasgger

Terms of service

default

GET /predict Predict if Customer would buy the product or not.

Parameters

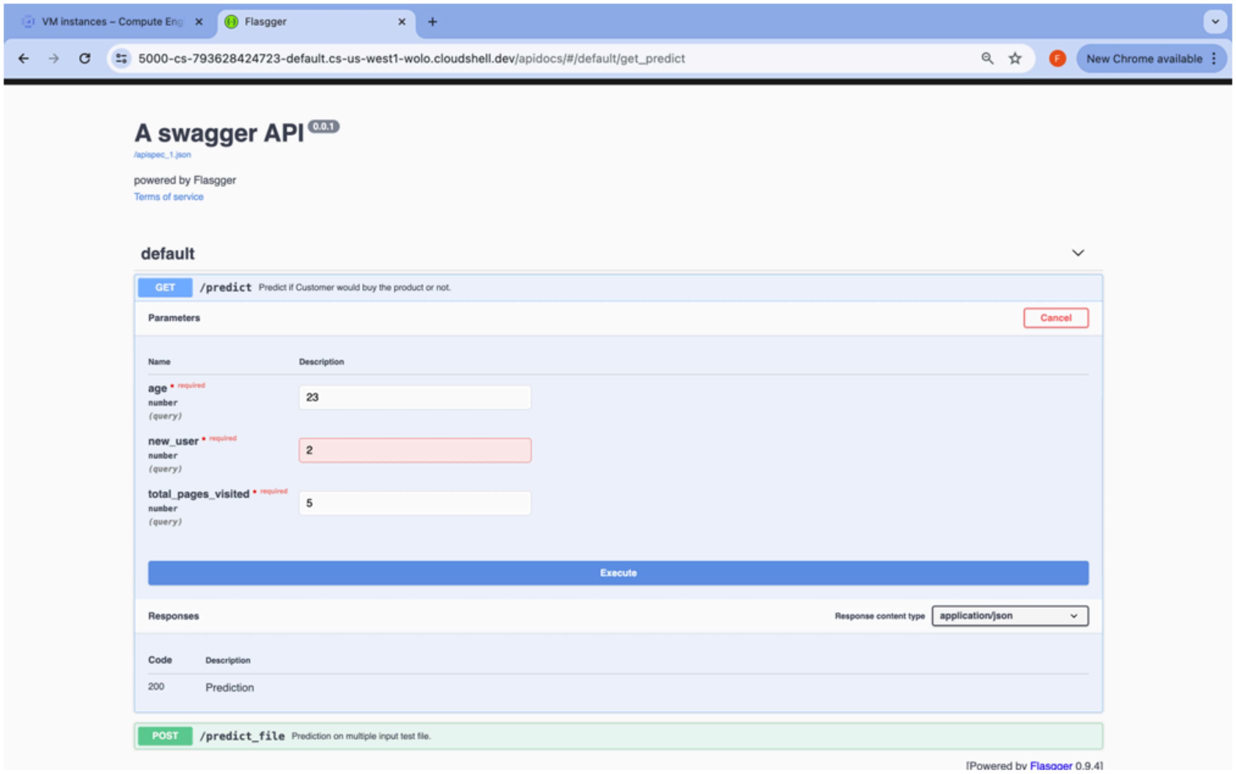
Name	Description
age * required number (query)	age
new_user * required number (query)	new_user
total_pages_visited * required number (query)	total_pages_visited

Responses

Response content type: application/json

Code	Description
200	Prediction

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



5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/apidocs/#/default/get_predict

A swagger API 0.0.1

/apispec_1.json

powered by Flasgger

Terms of service

default

GET /predict Predict if Customer would buy the product or not.

Parameters

Name	Description
age * required number (query)	23
new_user * required number (query)	2
total_pages_visited * required number (query)	5

Responses

Response content type: application/json

Code	Description
200	Prediction

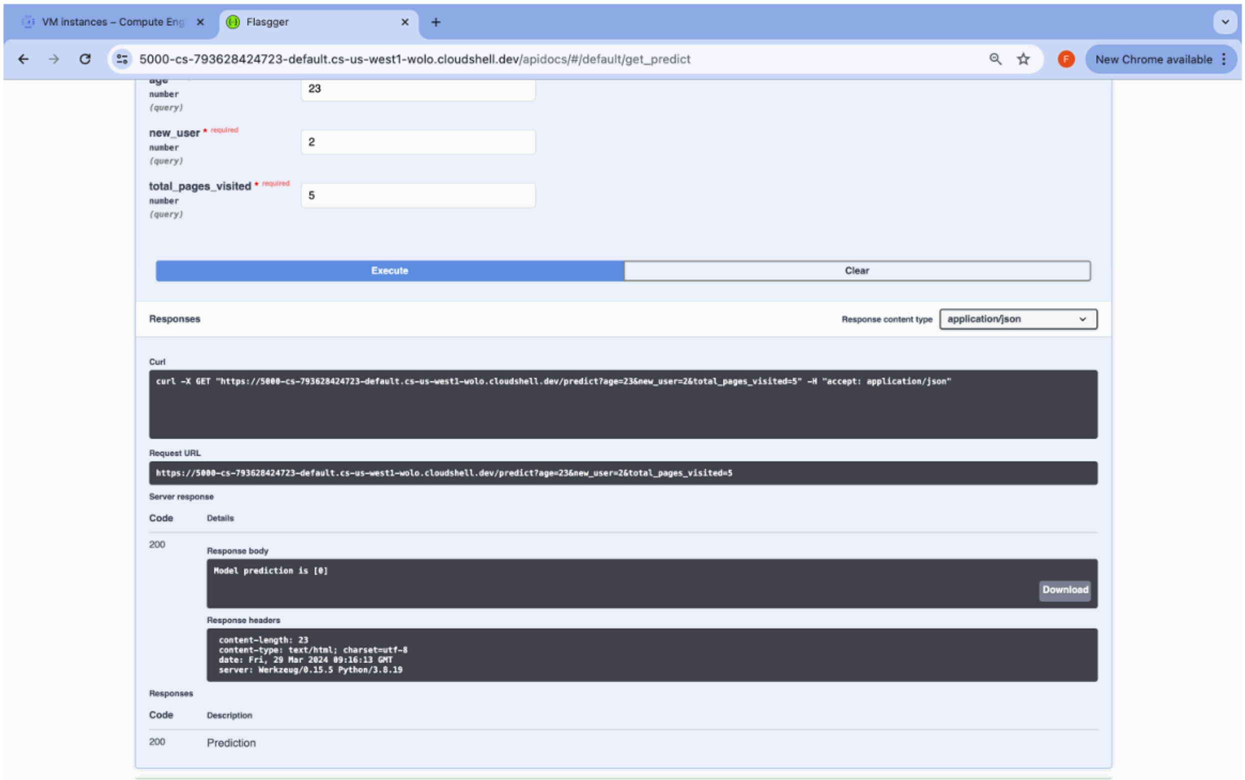
POST /predict_file Prediction on multiple input test file.

[Powered by Flasgger 0.9.4]

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

Machine Learning on Kubernetes

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



VM instances – Compute Eng
Flasgger
5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/apidocs/#/default/get_predict
New Chrome available

age
number
(query)
23

new_user * required
number
(query)
2

total_pages_visited * required
number
(query)
5

Responses

Response content type: application/json

Curl

```
curl -X GET "https://5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/predict?age=23&new_user=2&total_pages_visited=5" -H "accept: application/json"
```

Request URL

```
https://5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/predict?age=23&new_user=2&total_pages_visited=5
```

Server response

Code	Details
200	Response body

```
Model prediction is [0]
```

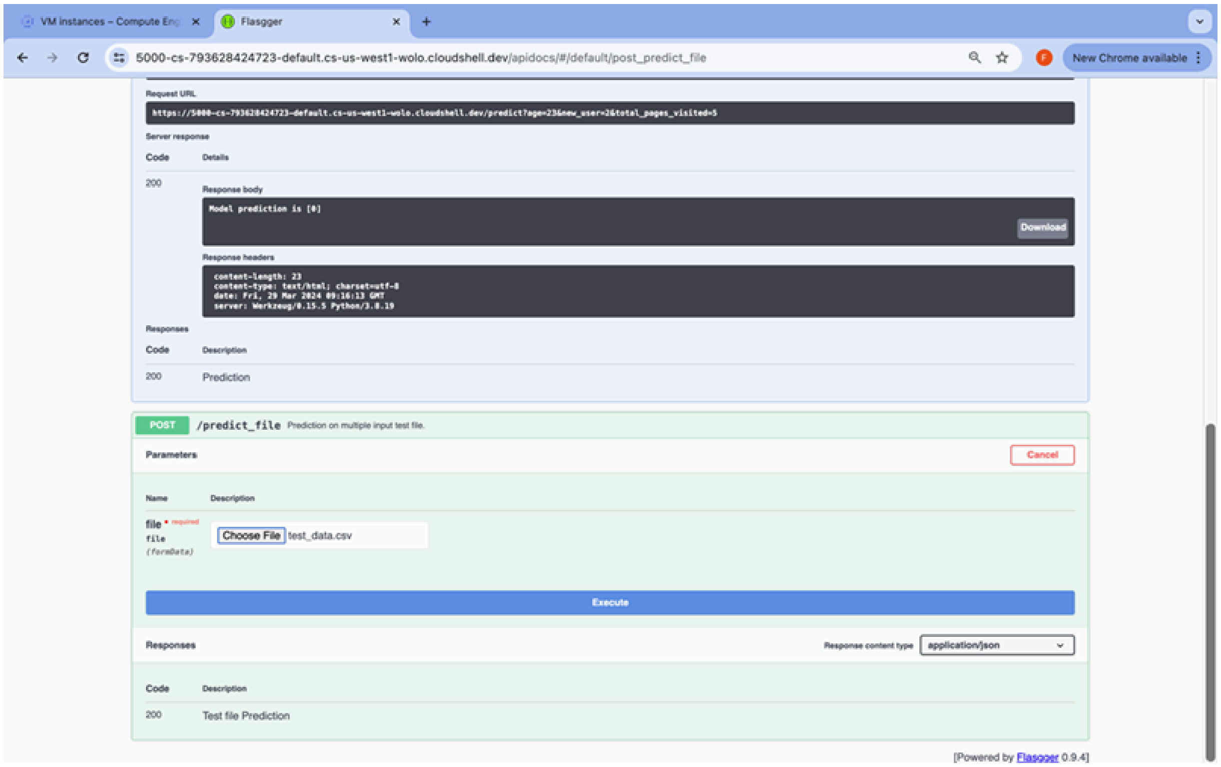
Response headers

```
content-length: 23
content-type: text/html; charset=utf-8
date: Fri, 29 Mar 2024 09:16:13 GMT
server: Werkzeug/0.15.5 Python/3.8.19
```

Responses

Code	Description
200	Prediction

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



5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/apidocs/#/default/post_predict_file

Request URL

```
https://5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/predict?age=23&new_user=2&total_pages_visited=5
```

Server response

Code	Details
200	Response body `Model prediction is [0]` Response headers

```
content-length: 23
content-type: text/html; charset=utf-8
date: Fri, 29 Mar 2024 09:16:13 GMT
server: Werkzeug/0.15.5 Python/3.8.19
```

Responses

Code	Description
200	Prediction

POST /predict_file Prediction on multiple input test file.

Parameters

Name	Description
file * required file (formData)	Choose File test_data.csv

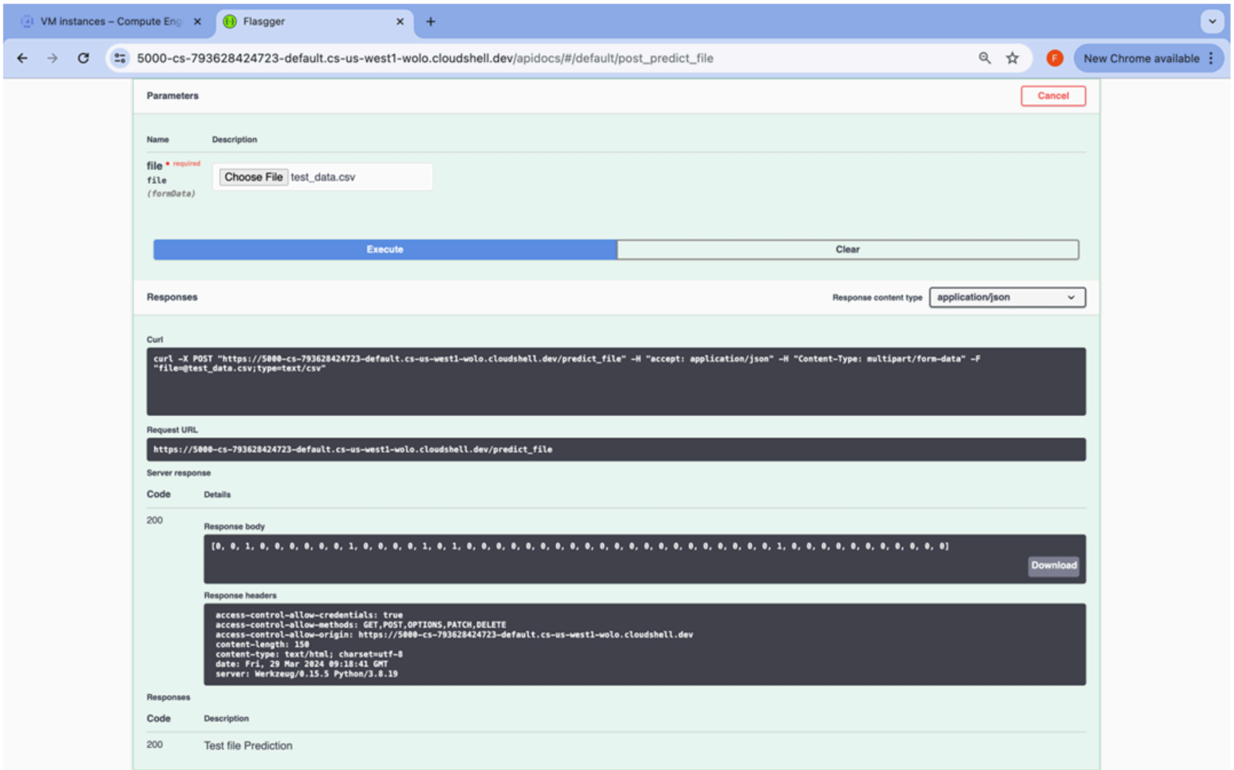
Responses — Response content type: application/json

Code	Description
200	Test file Prediction

[Powered by Flasgger 0.9.4]

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.

Machine Learning on Kubernetes



file: test_data.csv

Response content type: application/json

Curl

```
curl -X POST "https://5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/predict_file" -H "accept: application/json" -H "Content-Type: multipart/form-data" -F "file=@test_data.csv;type=text/csv"
```

Request URL

```
https://5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev/predict_file
```

Server response

Code	Details
200	Response body

```
[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

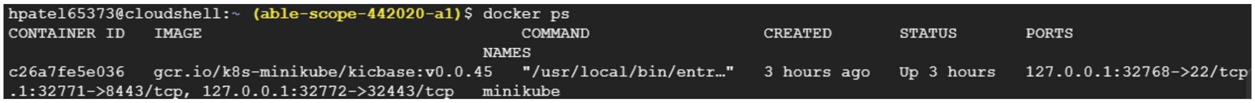
Response headers

```
access-control-allow-credentials: true
access-control-allow-methods: GET,POST,OPTIONS,PATCH,DELETE
access-control-allow-origin: https://5000-cs-793628424723-default.cs-us-west1-wolo.cloudshell.dev
content-length: 150
content-type: text/html; charset=utf-8
date: Fri, 29 Mar 2024 09:18:41 GMT
server: Werkzeug/0.15.5 Python/3.8.19
```

Responses

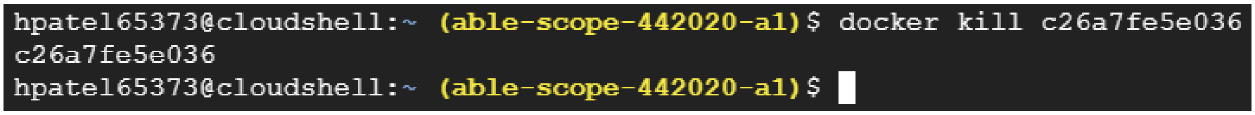
Code	Description
200	Test file Prediction

6. Stopping/killing the running container
    1. Use docker ps to list running Docker containers



```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker ps
CONTAINER ID   IMAGE                                 COMMAND                  CREATED       STATUS        PORTS
                                                NAMES
c26a7fe5e036   gcr.io/k8s-minikube/kicbase:v0.0.45   "/usr/local/bin/entr…"   3 hours ago   Up 3 hours    127.0.0.1:32768->22/tcp
.1:32771->8443/tcp, 127.0.0.1:32772->32443/tcp   minikube
```

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



```
hpatel65373@cloudshell:~ (able-scope-442020-a1)$ docker kill c26a7fe5e036
c26a7fe5e036
hpatel65373@cloudshell:~ (able-scope-442020-a1)$
```

Updating Portfolio- GitHub link

Cloud-Computing/kubernetes at main · hpatel65373/Cloud-Computing