Creating and uploading necessary files in GCP- Cloud Shell Terminal

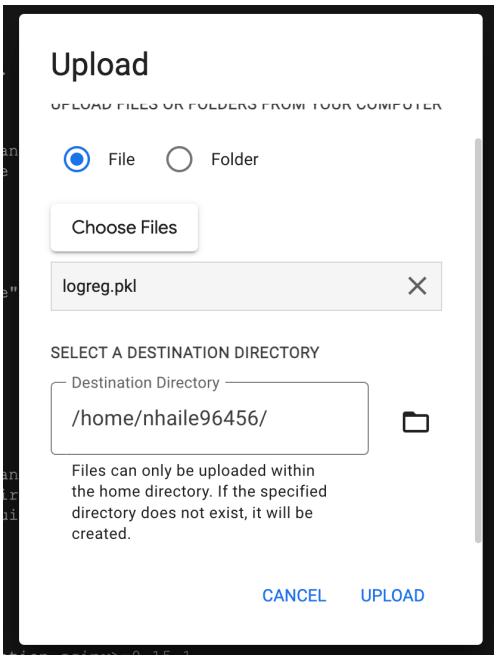
1. Start minikube in Google Cloud Platform
    - *minikube start*

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
* 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
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

2. Create a requirements.txt file.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi requirements.txt
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat requirements.txt
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
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

3. Upload logreg.pkl file to your workspace.



4. Create a flask_api.py file and paste the below python code in it.

```
from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
from flasgger import Swagger

app = Flask(__name__)
Swagger(app)

# Load the logistic regression model
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[0])

@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)
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi flask_api.py
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat flask_api.py
from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
from flasgger import Swagger

app = Flask(__name__)
Swagger(app)

# Load the logistic regression model
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():
    """
```

5. Create a DockerFile with the following content in it.

 # Use the official Python image from the Docker Hub
FROM python:3.8-slim
 # Set the working directory in the container
WORKDIR /app
 # Copy the current directory contents into the container at /app
COPY . /app
 # Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt
 # Make port 5000 available to the world outside this container
EXPOSE 5000
 # Define environment variable to prevent Python from writing .pyc files to disk
ENV PYTHONUNBUFFERED=1
 # Run flask_api.py when the container launches
CMD ["python", "flask_api.py"]

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi Dockerfile
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat Dockerfile
# Use the official Python image from the Docker Hub
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable to prevent Python from writing .pyc files to disk
ENV PYTHONUNBUFFERED=1

# Run flask_api.py when the container launches
CMD ["python", "flask_api.py"]

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Explanation

FROM python:3.8-slim: This line specifies the base image for the Docker container, which is a lightweight version of Python 3.8.
Working Directory:

WORKDIR /app: This sets the working directory to /app in the container. All subsequent commands will be run from this directory.
Copy Files:

COPY . /app: This copies the contents of the current directory on your host machine to the /app directory in the container.

Install Dependencies:

RUN pip install --no-cache-dir -r requirements.txt: This installs the dependencies listed in requirements.txt using pip. The --no-cache-dir option reduces the size of the image by not caching the downloaded packages.
Expose Port:

EXPOSE 5000: This makes port 5000 available for network connections. It does not actually publish the port; it simply serves as documentation.
Environment Variable:

ENV PYTHONUNBUFFERED=1: This prevents Python from buffering stdout and stderr, ensuring that logs are immediately available in Docker.
Command:

CMD ["python", "flask_api.py"]: This specifies the command to run the Flask application when the container starts.

6. To build the docker image use the command.

sudo docker build -t ml_app_docker .

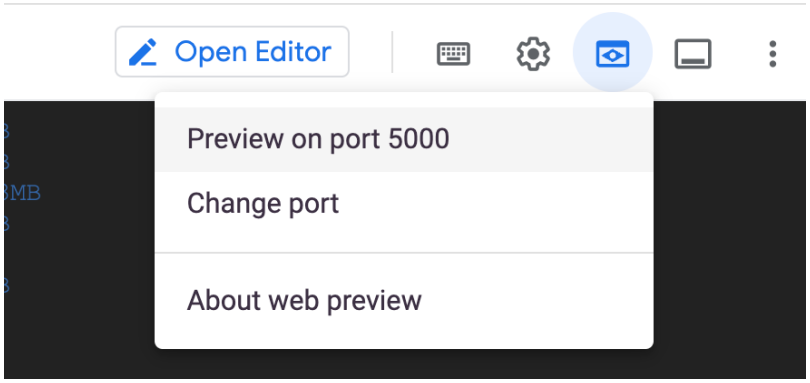
```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ sudo docker build -t ml_app_docker .
[+] Building 39.0s (9/9) FINISHED                                                                                     docker:default
 => [internal] load build definition from Dockerfile                                                                            0.0s
 => => transferring dockerfile: 621B                                                                                            0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                                                              0.7s
 => [internal] load .dockerignore                                                                                               0.0s
 => => transferring context: 2B                                                                                                 0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:c177b5b444d6913678d80bd26af131187de166cd68ac66605acbbc76e1b343d7        4.4s
 => => resolve docker.io/library/python:3.8-slim@sha256:c177b5b444d6913678d80bd26af131187de166cd68ac66605acbbc76e1b343d7        0.0s
 => => sha256:fea4f2170757fa9e87b6421086bf5d32880bd009555941641108ea89519b5742 11.67MB / 11.67MB                                0.6s
 => => sha256:c177b5b444d6913678d80bd26af131187de166cd68ac66605acbbc76e1b343d7 10.41kB / 10.41kB                                0.0s
 => => sha256:b2d6266a63eafff4fcee6a9149b80686ff6c17b0e4556d0320198ce85e9aa41d 1.94kB / 1.94kB                                  0.0s
 => => sha256:5208c64e1783e15b2d0ee357a3979688ae3faa21c6a43f4965e3530d68abbcfc 6.93kB / 6.93kB                                  0.0s
 => => sha256:efc2b5ad9eec05befa54239d53feeae3569ccbef689aa5e5dbfc25da6c4df559 29.13MB / 29.13MB                                0.5s
 => => sha256:0d935f02ede5b557e3899b4161a3a7f7dd8461fd62d558451b3884172a710962 3.51MB / 3.51MB                                  0.4s
 => => sha256:e5635d0cdd4c514a4e76d97174785a452a1e81b87b6a8bd8ce09c5ac2d135df8 230B / 230B                                      0.6s
 => => sha256:ebe530eb534fda723884e0b61fa4633bb792a9600f452a779301c8a7e4216d83 2.78MB / 2.78MB                                  0.8s
 => => extracting sha256:efc2b5ad9eec05befa54239d53feeae3569ccbef689aa5e5dbfc25da6c4df559                                       2.2s
 => => extracting sha256:0d935f02ede5b557e3899b4161a3a7f7dd8461fd62d558451b3884172a710962                                       0.2s
 => => extracting sha256:fea4f2170757fa9e87b6421086bf5d32880bd009555941641108ea89519b5742                                       0.6s
 => => extracting sha256:e5635d0cdd4c514a4e76d97174785a452a1e81b87b6a8bd8ce09c5ac2d135df8                                       0.0s
 => => extracting sha256:ebe530eb534fda723884e0b61fa4633bb792a9600f452a779301c8a7e4216d83                                       0.3s
 => [internal] load build context                                                                                               1.5s
 => => transferring context: 62.48MB                                                                                            1.4s
 => [2/4] WORKDIR /app                                                                                                          1.1s
 => [3/4] COPY . /app                                                                                                           0.5s
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt                                                                   29.0s
 => exporting to image                                                                                                          3.2s
 => => exporting layers                                                                                                         3.2s
 => => writing image sha256:e0e1b675538fcdc838c47bcc32929fdeaf6aeaeb562e2d22a848eec4207e97e4                                    0.0s
 => => naming to docker.io/library/ml_app_docker                                                                                0.0s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

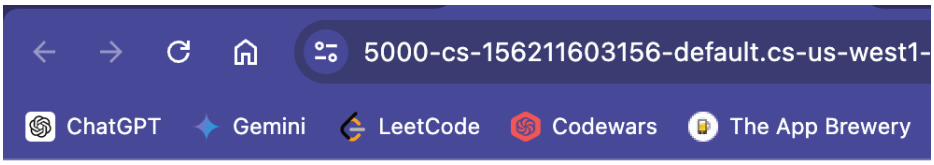
7. The following command runs a Docker container from the ml_app_docker image.

docker container run -p 5000:5000 ml_app_docker

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker container run -p 5000:5000 ml_app_docker
/usr/local/lib/python3.8/site-packages/sklearn/base.py:310: UserWarning: Trying to unpickle estimator LogisticRegression from version 0.23.2 when using version 0.24.2.
This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
 * Serving Flask app "flask_api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
/usr/local/lib/python3.8/site-packages/sklearn/base.py:310: UserWarning: Trying to unpickle estimator LogisticRegression from version 0.23.2 when using version 0.24.2.
This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 269-833-583
```

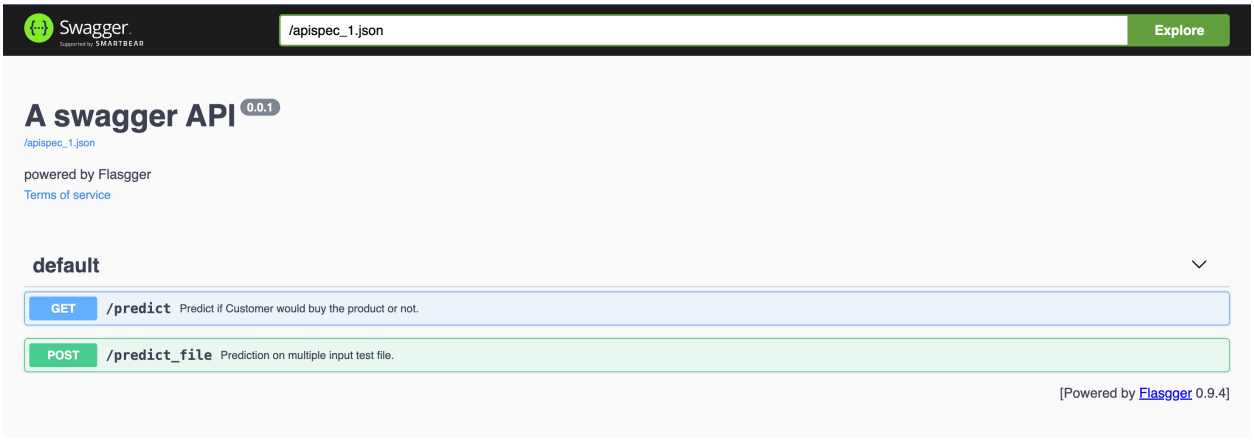
8. On the top right side of the screen, click the little eye shaped button. Change the port number if it's not already 5000. You will expect to see some sort of a welcome message.



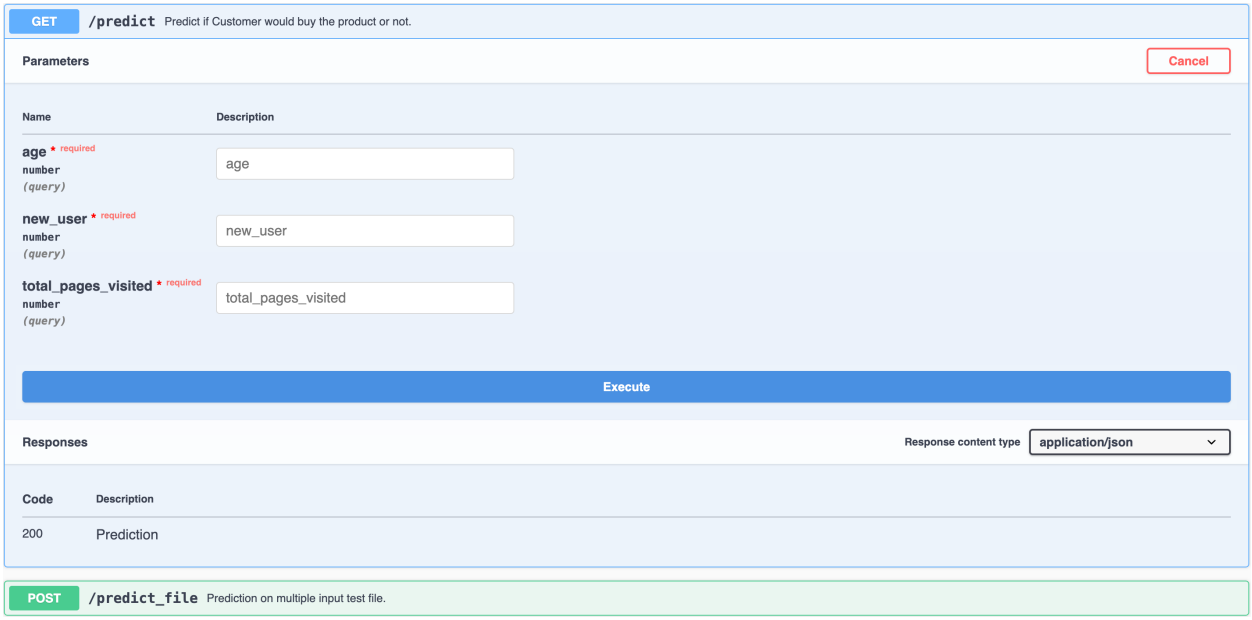


Welcome to the Flask API!

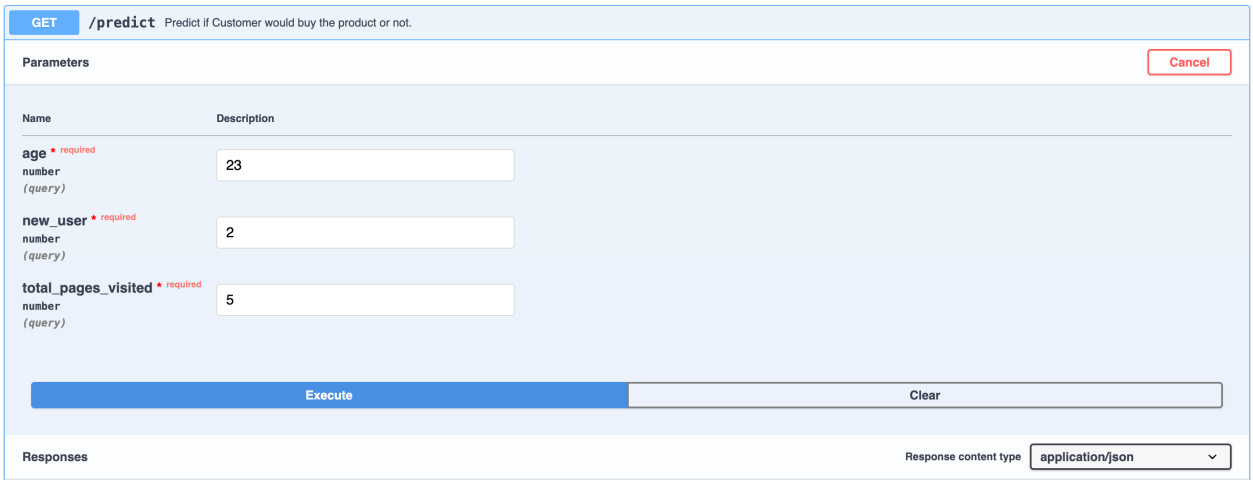
9. Add /apidocs/ at the end of the URL and you will see the home page of Swagger-UI.



10. Click ‘Get’ and ‘Try it out’ at the top right side to get the following page.

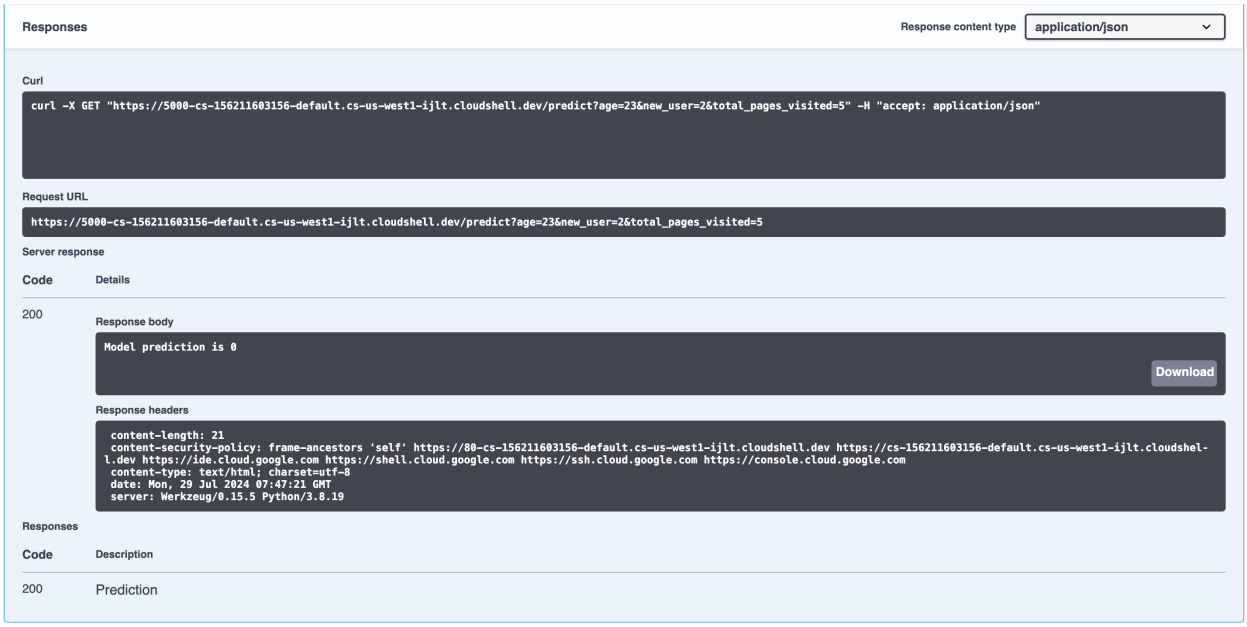


11. Enter values for the input parameters and click ‘Execute.’

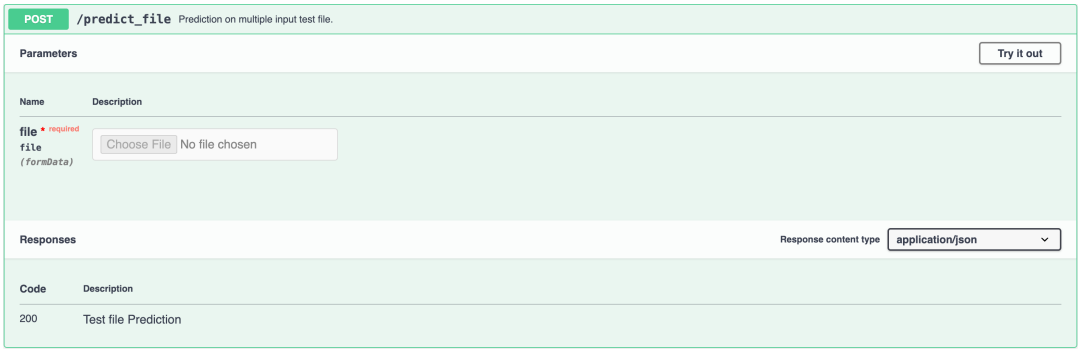


12. Upon the execution call, the request goes to the app and predictions are made by the model.

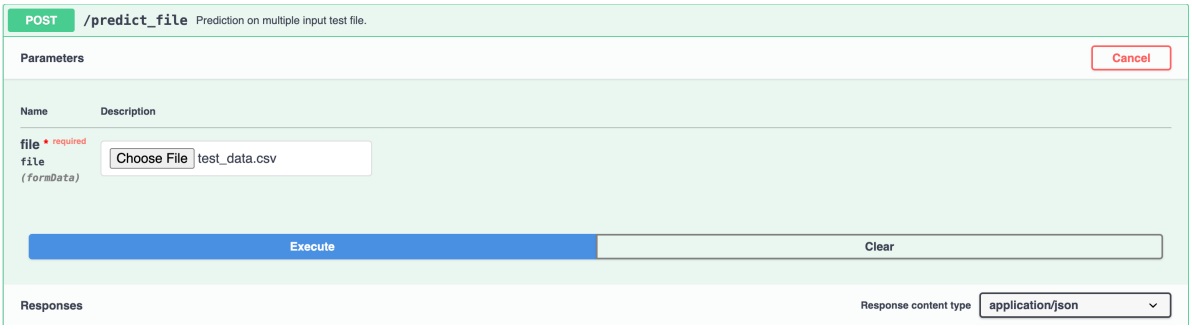
The result of the model prediction is displayed in the Prediction section of the page as following



13. Next, the app can make predictions for a group of customers (test data) by clicking 'Post'.



14. Upload the test_data.csv file and click 'Execute.'



15. The model would make the predictions, and the results would be displayed as shown below.

Curl

```
curl -X POST "https://5000-cs-156211603156-default.cs-us-west1-ijlt.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-156211603156-default.cs-us-west1-ijlt.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, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Download

Response headers

```
 access-control-allow-credentials: true
 access-control-allow-methods: GET,POST,OPTIONS,PATCH,DELETE
 access-control-allow-origin: https://5000-cs-156211603156-default.cs-us-west1-ijlt.cloudshell.dev
 content-length: 150
 content-security-policy: frame-ancestors 'self' https://80-cs-156211603156-default.cs-us-west1-ijlt.cloudshell.dev https://cs-156211603156-default.cs-us-west1-ijlt.cloudshel-
l.dev https://ide.cloud.google.com https://shell.cloud.google.com https://ssh.cloud.google.com https://console.cloud.google.com
 content-type: text/html; charset=utf-8
 date: Mon, 29 Jul 2024 07:52:31 GMT
 server: Werkzeug/0.15.5 Python/3.8.19
```

Responses

Code	Description
200	Test file Prediction

16. To list running docker containers to kill them, run the command docker ps.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker ps
CONTAINER ID   IMAGE                                 COMMAND                  CREATED          STATUS          PORTS
                                                                               NAMES
6393b5a57ff9   gcr.io/k8s-minikube/kicbase:v0.0.44   "/usr/local/bin/entr…"   39 minutes ago   Up 39 minutes   127.0.0.1:32768->22/tcp, 127.0.0.1:32769->2376/tcp, 127.0
.0.1:32770->5000/tcp, 127.0.0.1:32771->8443/tcp, 127.0.0.1:32772->32443/tcp   minikube
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

17. Then run ‘*docker kill <container_id>*’.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker kill 6393b5a57ff9
6393b5a57ff9
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```