Machine Learning on Kubernetes

Natnael Haile
ID: 20007
July 18, 2024

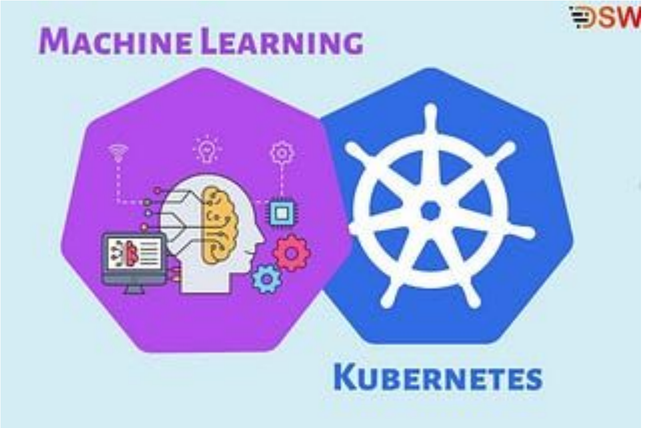


Table of Contents

Introduction

- **Project Overview**
  - This project demonstrates the deployment of a machine learning model using Flask API on a Kubernetes cluster.
  - The model predicts customer behavior based on input features using logistic regression.
- **Technologies Used**
  - **Google Cloud Platform (GCP):** Provides the infrastructure for running Kubernetes.
  - **Kubernetes:** Manages containerized applications in a clustered environment.
  - **Docker:** Containerizes the Flask application and its dependencies.
  - **Flask:** A lightweight web framework for building the API.
  - **Python:** The programming language used for the model and API implementation.
  - **Minikube:** Local Kubernetes cluster setup tool used for development and testing.

Design: Main Components

- **Flask API:** Handles incoming requests, loads the pre-trained machine learning model, and returns predictions.
- **Docker Image:** Contains the Flask application and its dependencies.
- **Swagger-UI:** Provides an interactive interface for testing the API endpoints.

Design: Project Workflow

- **Model Training:** Logistic regression model trained on customer data.
- **API Development:** Flask API developed to serve model predictions.
- **Containerization:** Flask application containerized using Docker.
- **Deployment:** Docker container deployed on a Kubernetes cluster using Minikube.
- **Testing:** API endpoints tested using Swagger-UI to ensure correct functionality.

Implementation: Environment Setup

- Start minikube.

```
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)$
```

Implementation: Environment Setup

- **Create requirements.txt.**
  - Start Minikube in Google Cloud Platform to create a local Kubernetes cluster.

```
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)$
```

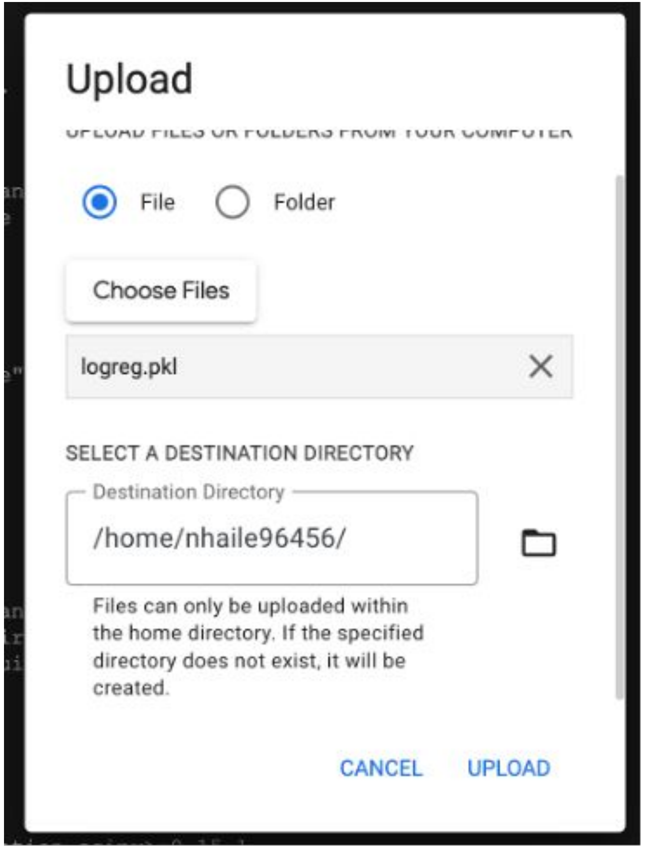
Implementation: Environment Setup

- **Create requirements.txt.**
    - List the necessary Python packages required for the project.

```
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)$
```

Implementation: Environment Setup

- **Upload Model**
  - Upload the pre-trained logistic regression model (logreg.pkl) to the working directory.



Implementation: Environment Setup

- **Develop Flask API**
  - Create `flask_api.py` to handle API requests and return model predictions.

```
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():
    """
```

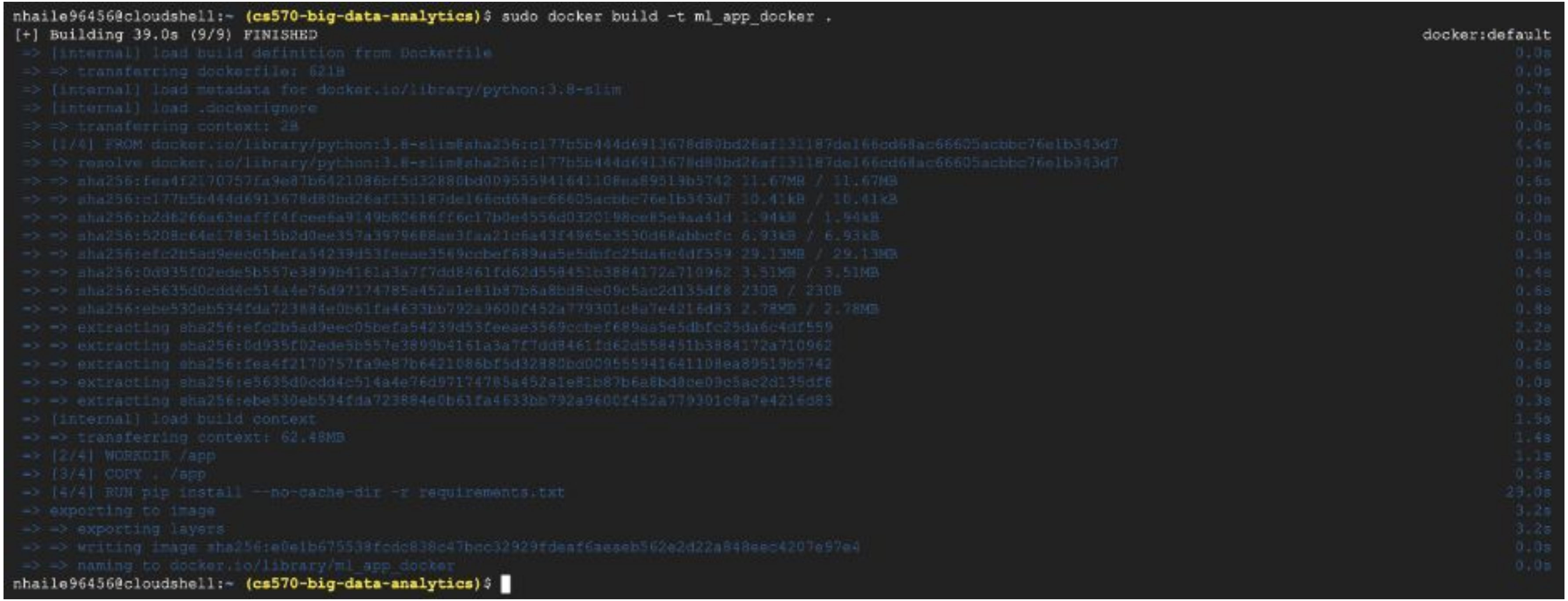
Implementation: Environment Setup

- **Create Dockerfile**
  - Define the Dockerfile to containerize the Flask application and its dependencies.

```
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)$
```

Implementation: Environment Setup

- **Build Docker Image**
  - Use Docker to build an image from the Dockerfile.
    - sudo docker build -t ml_app_docker .



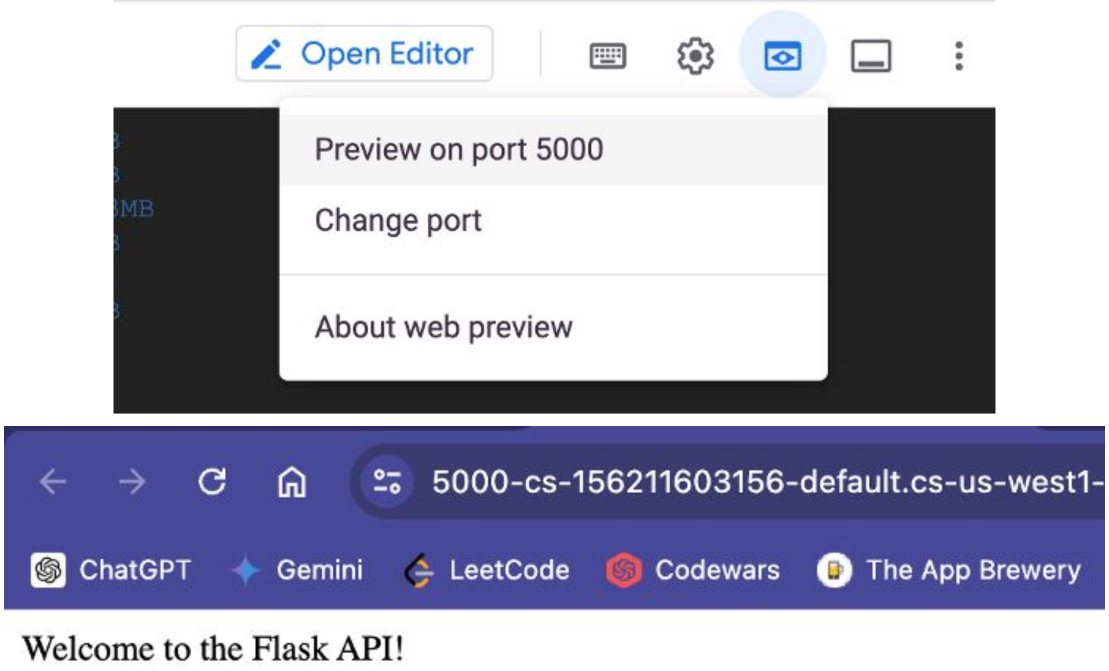
Implementation: Environment Setup

- **Run Docker Container**
  - Run the Docker container, exposing the Flask API on port 5000.
    - 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
```

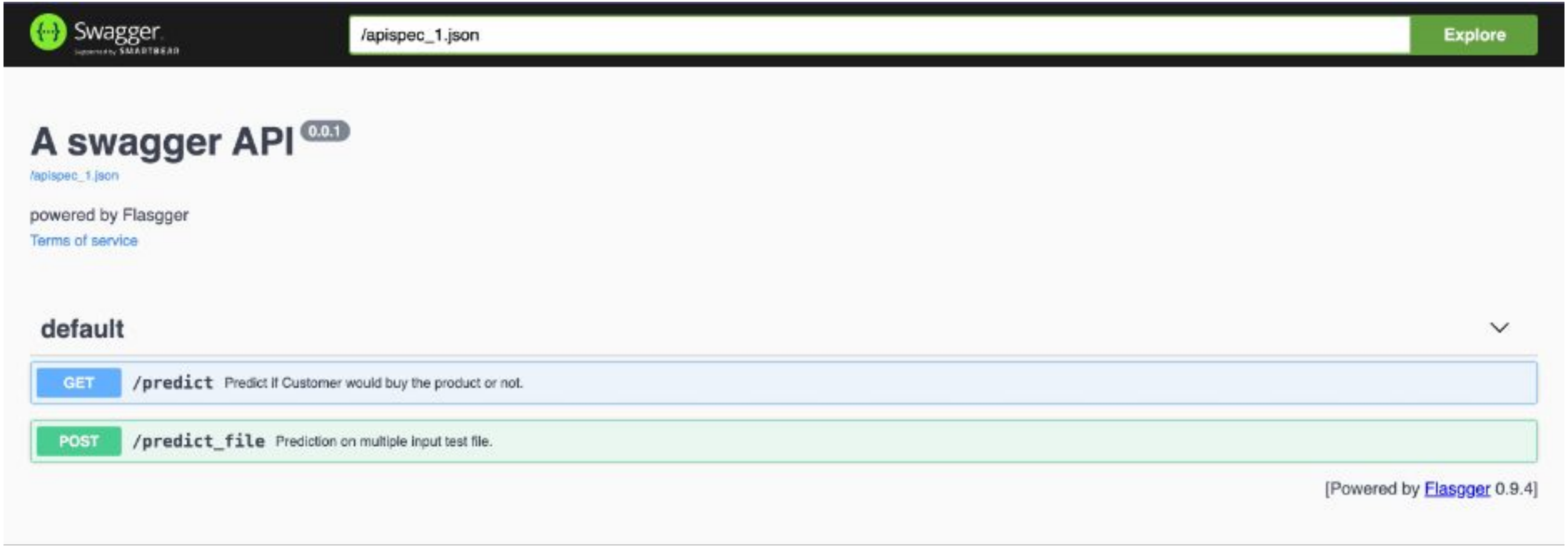
Implementation: Environment Setup

- **Expose Port in Cloud Shell**
  - Configure port forwarding in Google Cloud Shell to access the Flask API.



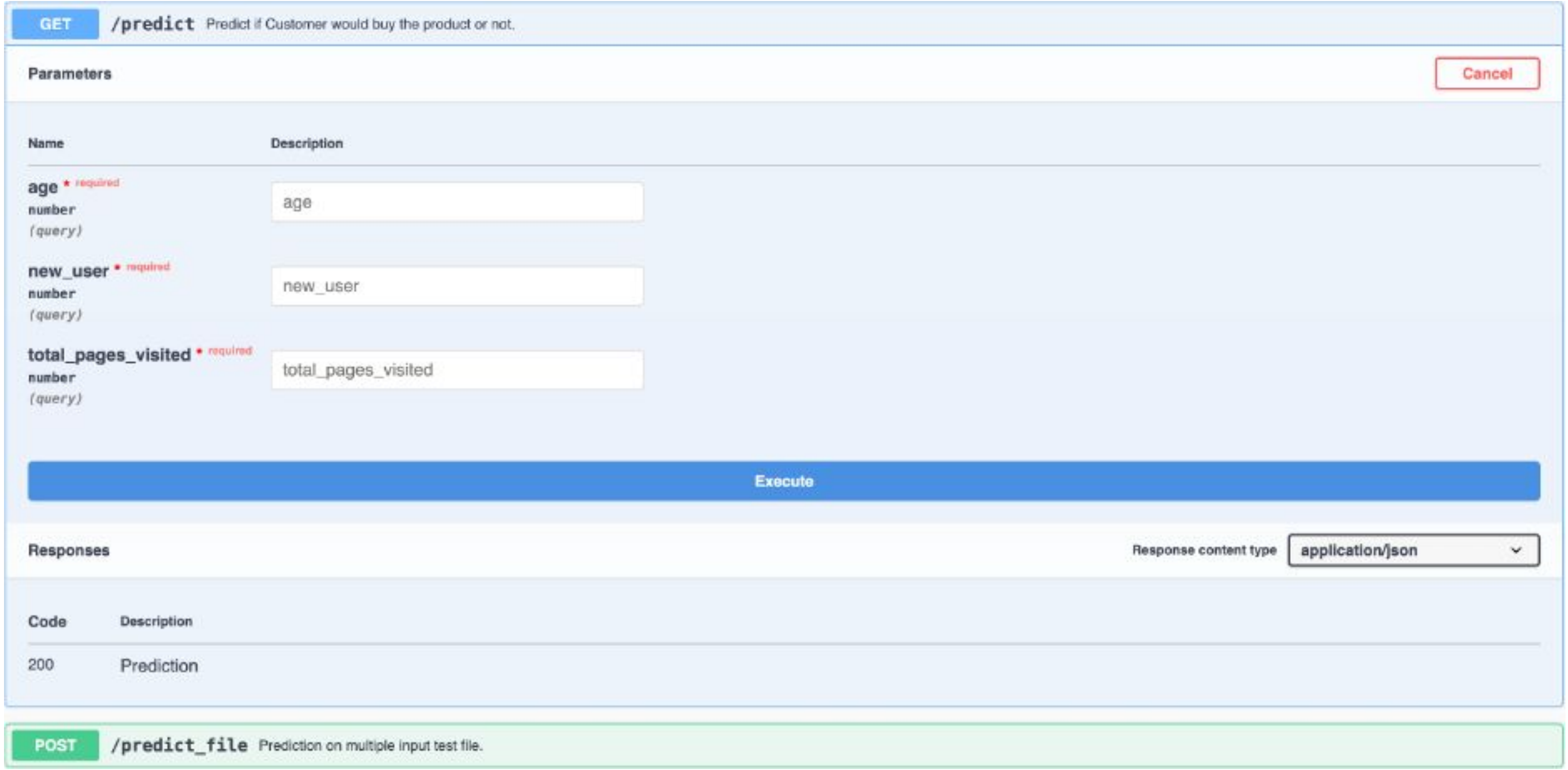
Implementation: Environment Setup

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



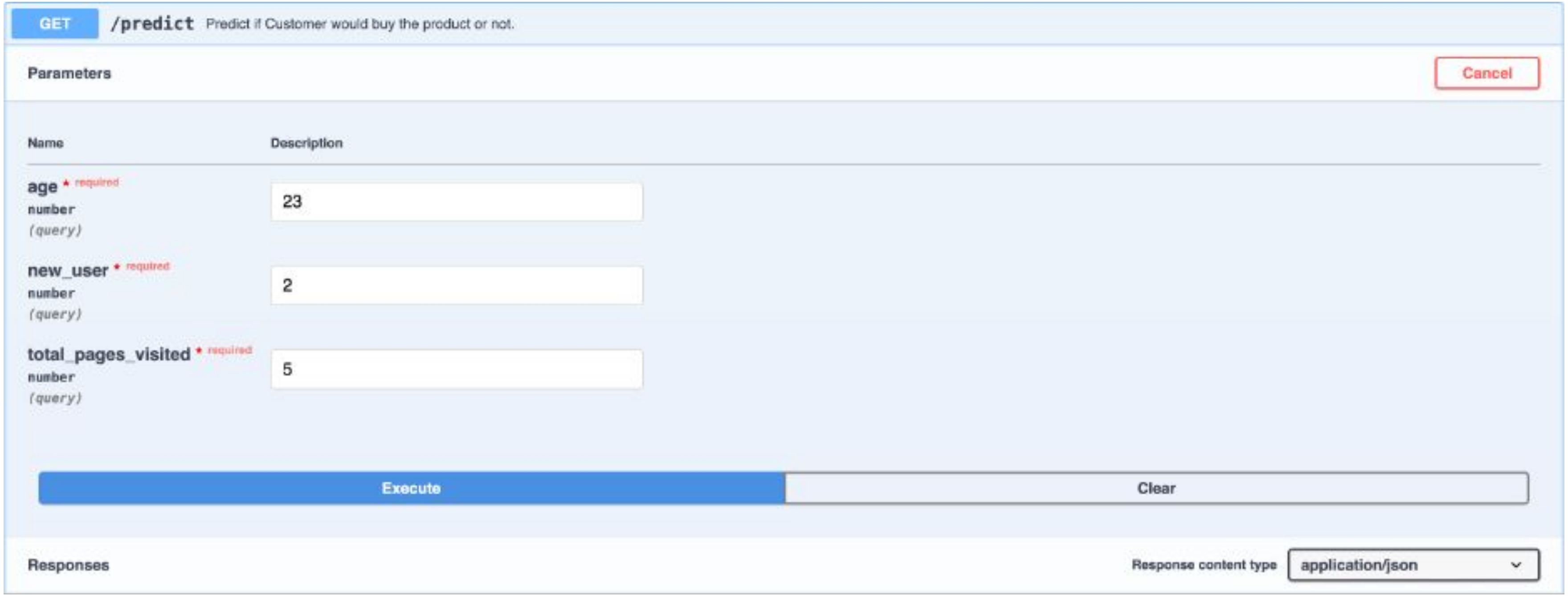
Test: GET

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



Test: GET

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



Test: GET

- 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

Responses

Response content type: application/json

Curl

```
curl -X GET "https://5000-cs-156211603156-default.cs-us-west1-ijlt.cloudshell.dev/predict?age=23&new_user=2&total_pages_visited=5" -H "accept: application/json"
```

Request URL

```
https://5000-cs-156211603156-default.cs-us-west1-ijlt.cloudshell.dev/predict?age=23&new_user=2&total_pages_visited=5
```

Server response

Code	Details
200	Response body

```
Model prediction is 0
```

Download

Response headers

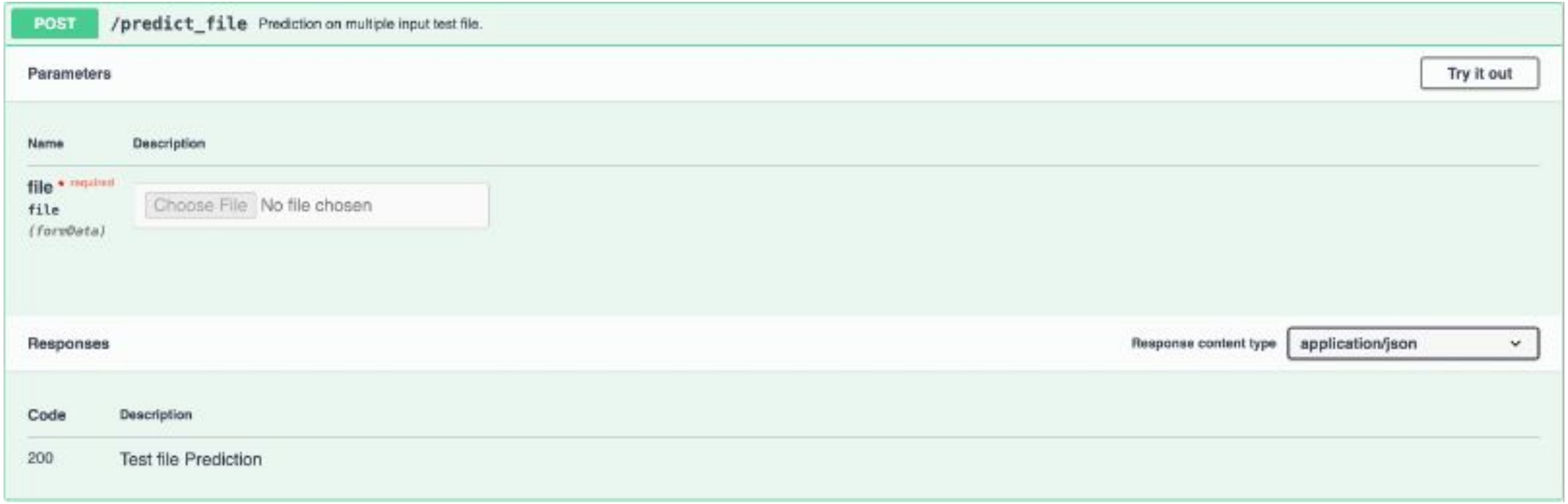
```
content-length: 21
content-security-policy: frame-ancestors 'self' https://80-cs-156211603156-default.cs-us-west1-ijlt.cloudshell.dev https://cs-156211603156-default.cs-us-west1-ijlt.cloudshel-l.dev https://ide.cloud.google.com https://shell.cloud.google.com https://ssh.cloud.google.com https://console.cloud.google.com
content-type: text/html; charset=utf-8
date: Mon, 29 Jul 2024 07:47:21 GMT
server: Werkzeug/0.15.5 Python/3.8.19
```

Responses

Code	Description
200	Prediction

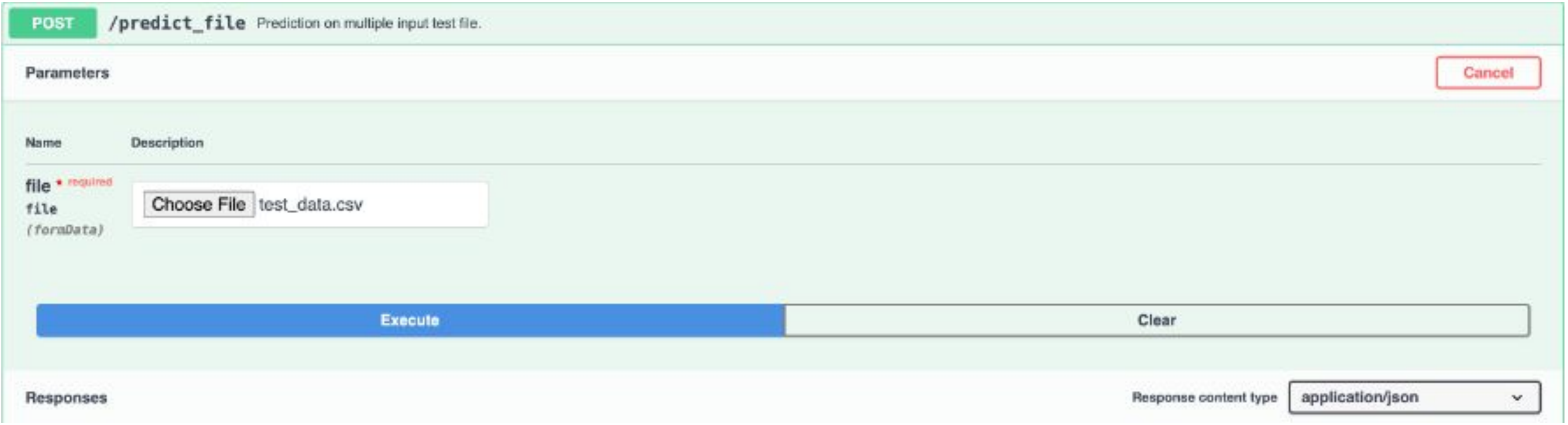
Test: POST

- **Next, the app can make predictions for a group of customers (test data) by clicking ‘Post’.**



Test: POST

- **Upload the test_data.csv file and click 'Execute.'**



Test: POST

- **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, 1, 0, 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.cloudshell.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

Test: Kill the Docker Container

- **To list running docker containers to kill them, run the command docker ps.**
- **Then run ‘docker kill <container_id>’.**

```
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)$
```

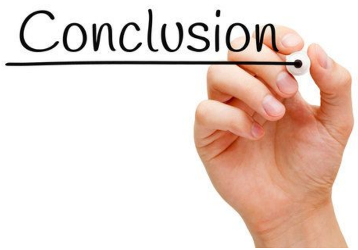
```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker kill 6393b5a57ff9
6393b5a57ff9
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Enhancement Ideas

- Integrate CI/CD pipelines to automate testing, building, and deploying updates to the Kubernetes cluster.
- Implement model versioning to manage and deploy multiple versions of the machine learning model, ensuring seamless updates and rollbacks.
- Enhance the scalability of the system using Kubernetes features like Horizontal Pod Autoscaler (HPA) to automatically scale the number of pods based on demand.
- Integrate advanced monitoring and logging solutions, such as Prometheus and Grafana, to gain deeper insights into application performance and detect issues proactively.
- Implement enhanced security measures, including role-based access control (RBAC) and network policies, to secure the application and its deployment environment.

Conclusion

- The machine learning model was successfully deployed on a Kubernetes cluster using Flask and Docker, demonstrating a robust and scalable architecture.
- A functional Flask API was developed to handle prediction requests, providing accurate and timely responses based on the logistic regression model.
- Swagger-UI was integrated for interactive documentation and testing, making it easy for users to test API endpoints and understand the available functionalities.
- The Flask application and its dependencies were efficiently containerized using Docker, ensuring consistency across different deployment environments.
- The project lays a solid foundation for future enhancements, including automated CI/CD pipelines, model versioning, and scalability improvements.



References

Machine Learning on Kubernetes | Data | eBook

Machine Learning on Kubernetes, published by packt

What is Kubernetes for MLOps

GitHub Link

- https://github.com/cur10usityDrives/Cloud-Computing/tree/main/Kubernetes/Machine-Learning


THANK YOU