Machine Learning Operations (MLOps)

Assignment 2

Task 4 – Model Deployment Using Cloud Services

Group Number 76

CHAUDHARI AAKASH VINAYAK (2022ac05607)
AATIF HUSSAIN WAZA (2022ac05405)
AJIT KUMAR YADAV (2022ac05720)
MOHAMMAD ZUBAIR (2022ac05121)

Contents

1. Overview

This document outlines the steps taken to deploy a trained model using Azure Functions, demonstrating how to create an API for making predictions. The deployment process utilized Visual Studio Code plugins to facilitate the deployment of the function app and local code to Azure.

2. Model Deployment Using Azure Functions

To deploy the model, Azure Functions was selected as the cloud service. This choice enables the creation of a serverless architecture, which allows for easy scaling and management.

Steps:

- Created an Azure Function that serves as the API endpoint for making predictions.
- Configured the function to accept HTTP POST requests with input data

Challenges:

- Issue: The model file (model.pkl) needed to be included in the deployment package.
- Solution: The model file was copied to the same directory as the Azure Function and included in the requirements.txt file to ensure all dependencies (like numpy, joblib) were installed on the cloud.

3. Setting Up the Environment

To deploy the Azure Function, the following tools and services were utilized:

- Azure Functions Core Tools: For local development and deployment.
- Visual Studio Code: For writing code and managing the deployment process.
- Azure CLI: For interacting with Azure services through the command line.

Steps:

1. Installed Azure Functions Core Tools and Visual Studio Code.
2. Created a new function app using Python as the runtime.

Challenge:

- **Issue**: Some functions weren't being recognized after deployment.

- **Solution**: Ensured that the correct Python environment and dependencies were specified in requirements.txt and redeployed the function.

4. Testing the API Endpoint

To ensure the deployment was successful, the endpoint was tested using `curl`.

CURL Command for Testing:

curl -X POST http://localhost:7071/ api/func_mlops_assignment_2 -H "Content-Type: application/json" -d '{

"sepal_length": 5.1,

"sepal_width": 3.5,

"petal_length": 1.4,

"petal_width": 0.2

}'

Replace *http://localhost:7071/ api/func_mlops_assignment_2* with the actual URL of your deployed function.

Expected Result:

The API should return a JSON response containing the prediction based on the input features.

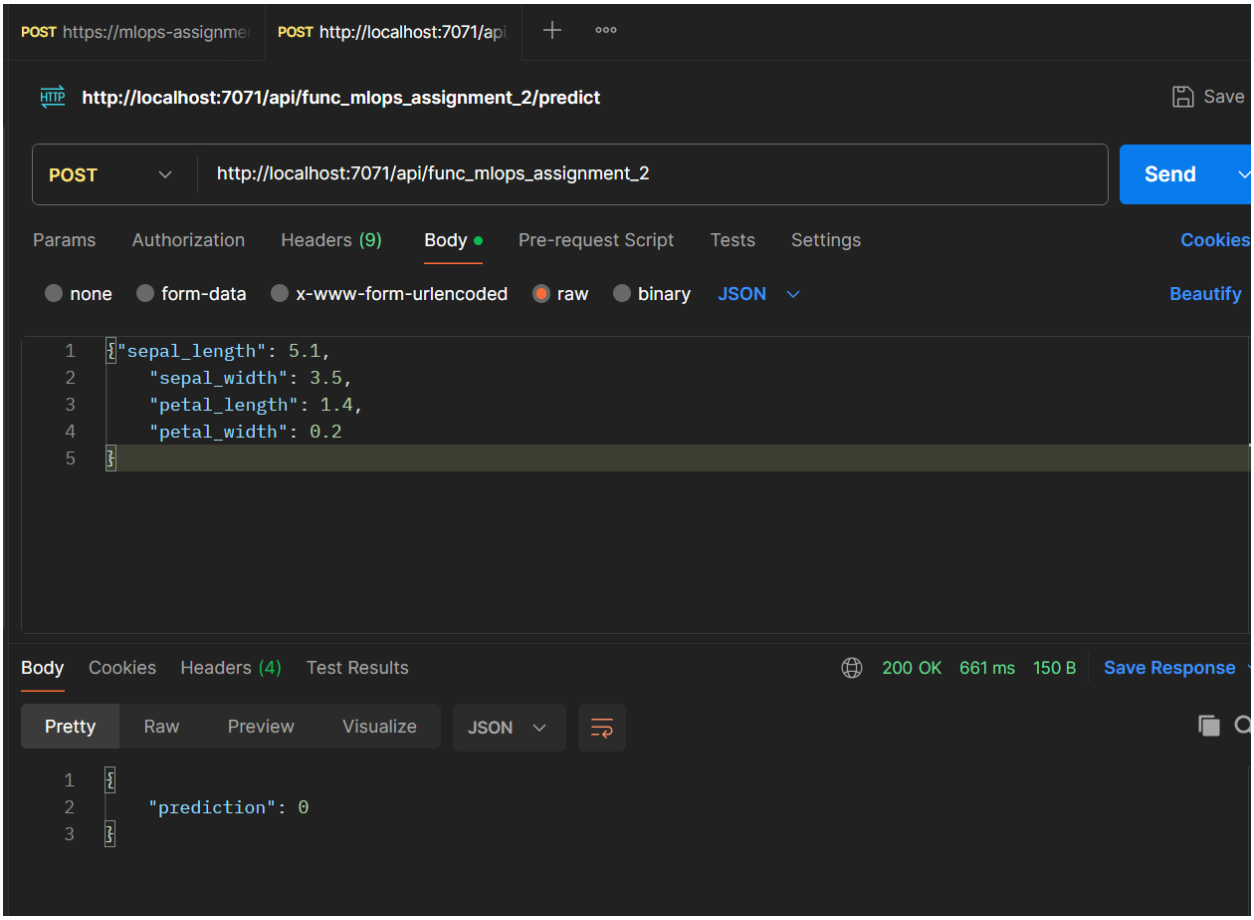

Figure 1 Function Local Testing

Challenge:

- **Issue**: Initially encountered CORS (Cross-Origin Resource Sharing) issues when testing from different environments.
- **Solution**: Configured the function app to handle CORS by setting up appropriate headers in Azure Functions.

5. Screenshots

Screenshots of the Azure portal showing the function app, the code in the Azure Functions, and deployment of code on Azure Functions.

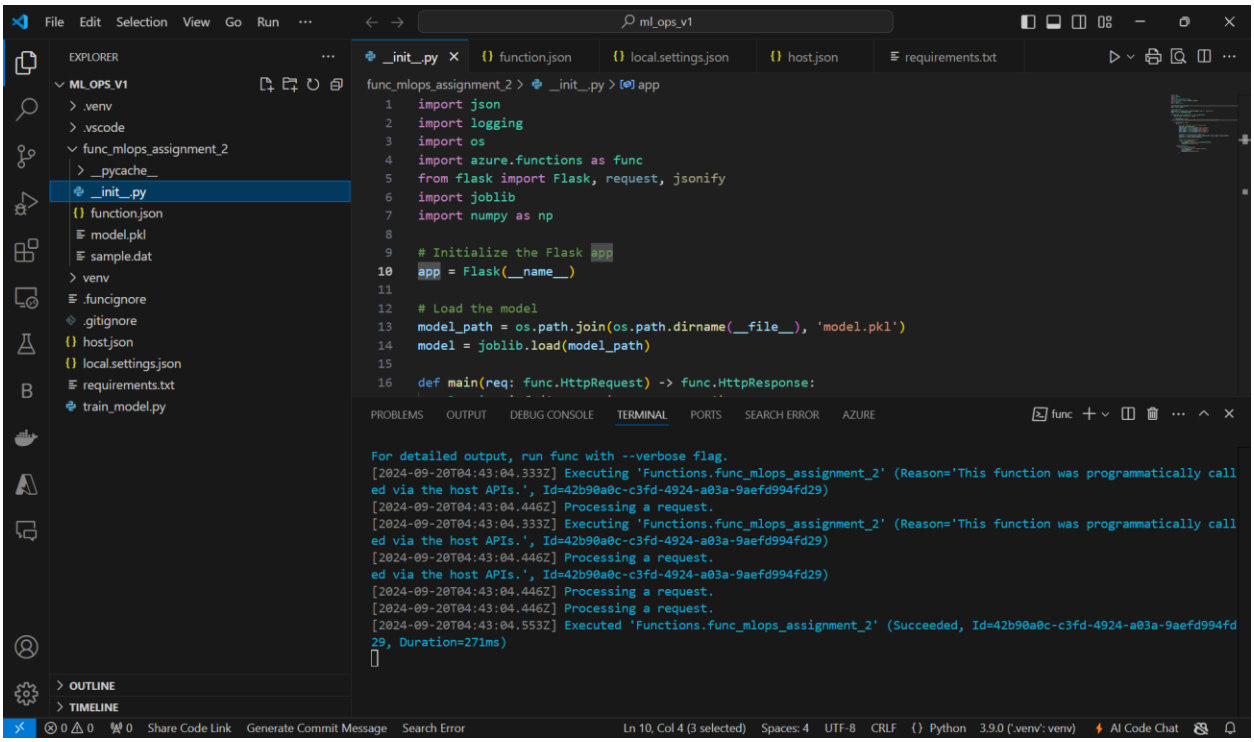

Figure 2 API code for Azure Functions

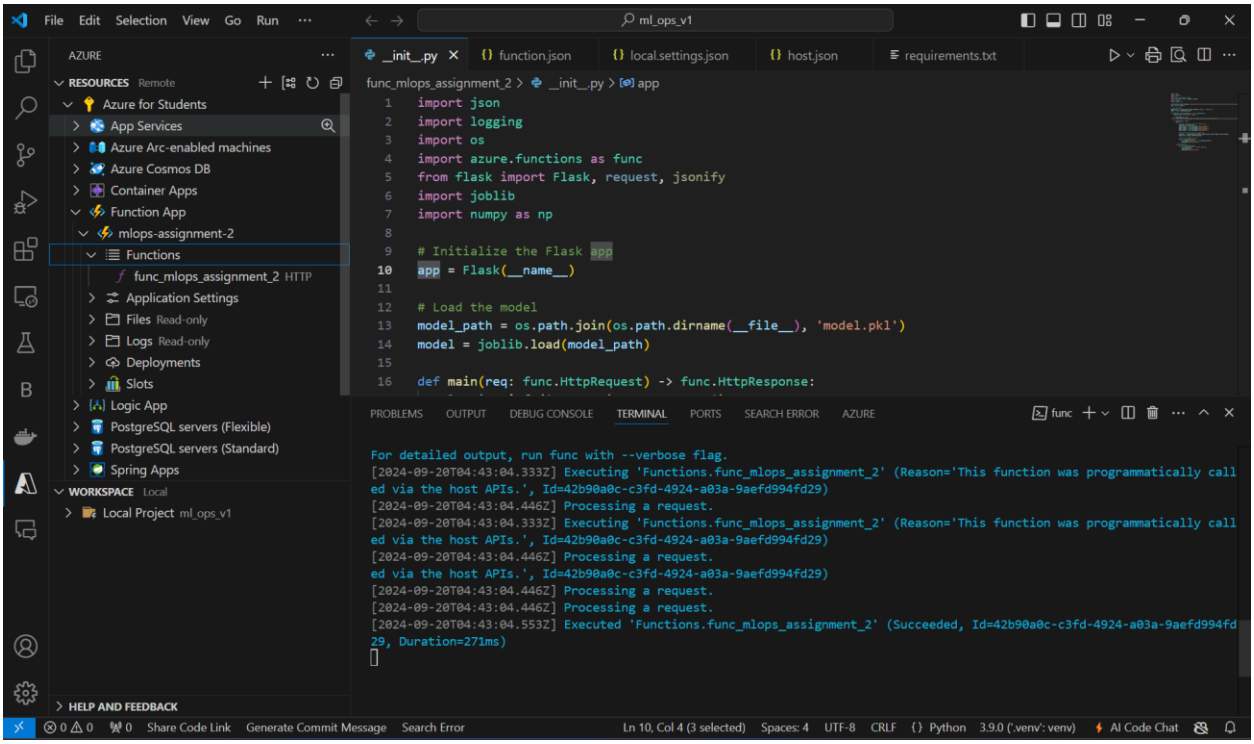

Figure 3 Creation of Azure Function App

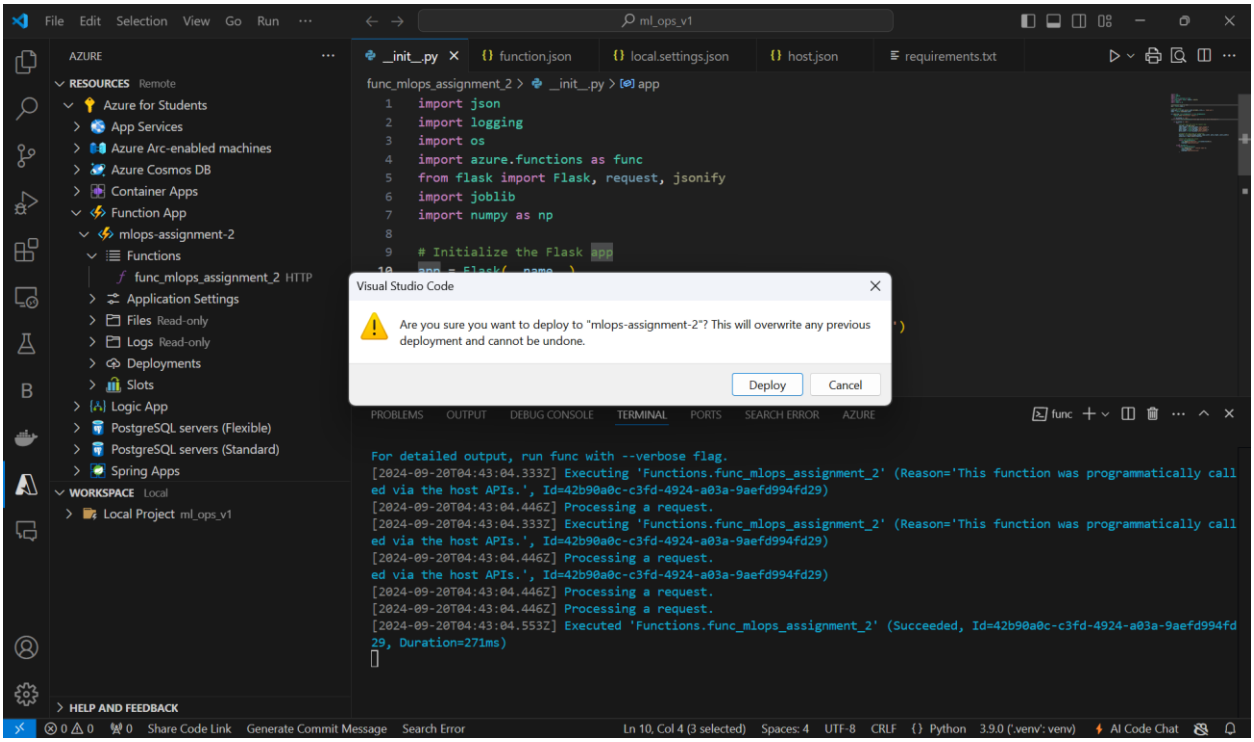

Figure 4 Deployment of Code API on Azure Function APP

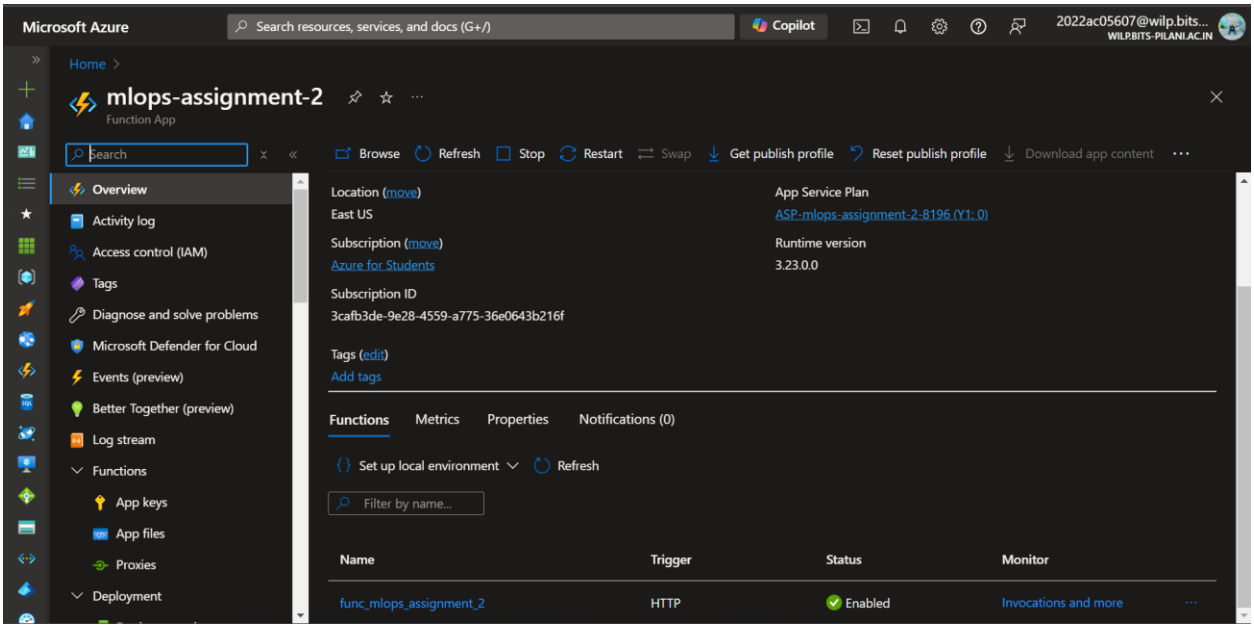

Figure 5 Created Azure Function App on Azure Portal

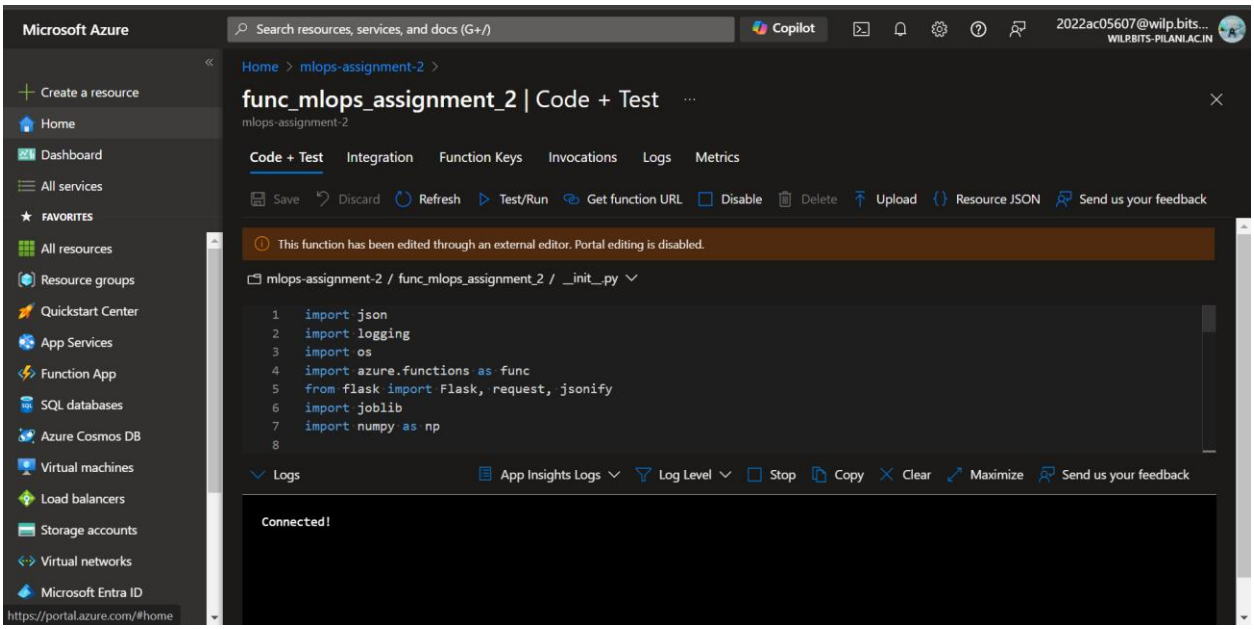

Figure 6 Deployed code output on Azure Portal

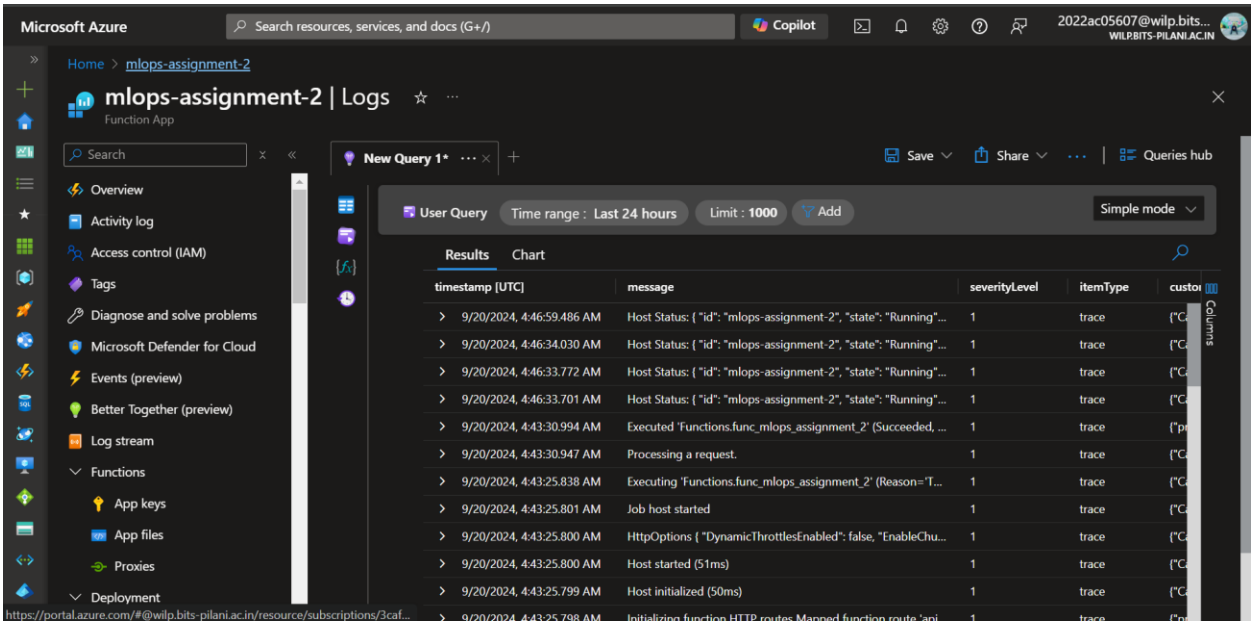

Figure 7 Logs of Azure Function App on Azure Portal

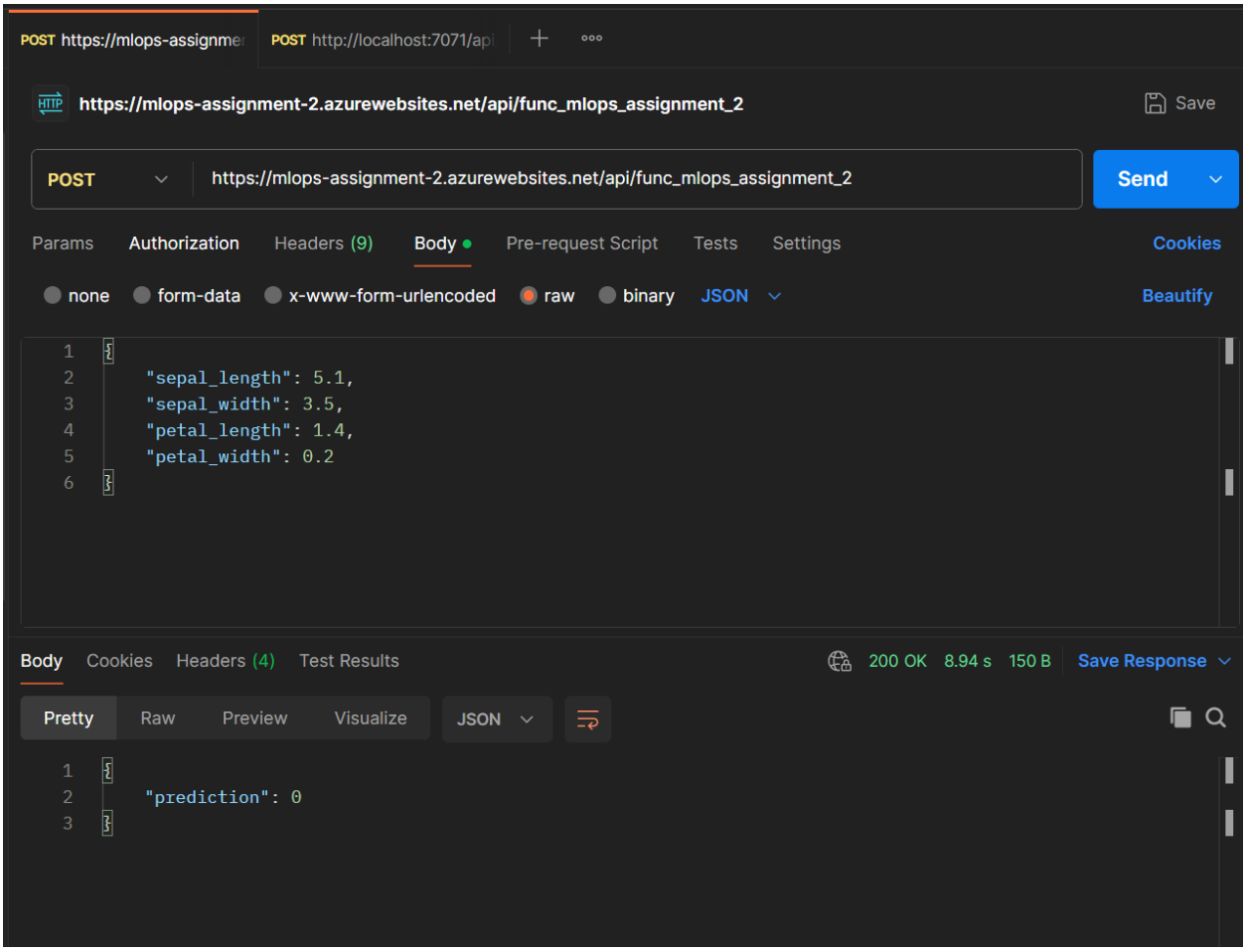

Figure 8 API call with the Azure Function App URL

6. Conclusion

The deployment of the machine learning model on Azure Functions, and the use of Visual Studio Code for development, proved to be a scalable and efficient method. The challenges related to CORS, and deployment were addressed successfully, and the API is now fully functional.

GitHub Link: https://github.com/AakashChaudhari03/MLOPS_ASSIGNMENT_2_GRP_NO_76