

Machine Learning Operations (MLOps)

Assignment 2

Task 5 –Workflow Documentation and Explanation

End-to-End Machine Learning Workflow with KizenML, XAI, and Cloud Deployment

Group Number 76

CHAUDHARI AAKASH VINAYAK (2022ac05607)

AATIF HUSSAIN WAZA (2022ac05405)

AJIT KUMAR YADAV (2022ac05720)

MOHAMMAD ZUBAIR (2022ac05121)

Contents

Project Overview	3
1. Data Collection and Preprocessing	3
2. Model Selection, Training, and Hyperparameter Tuning	4
3. Explainable AI (XAI) Implementation (6 Marks)	6
4. Model Deployment Using Cloud Services	7

Project Overview

This project demonstrates an end-to-end machine learning workflow, starting with data collection and preprocessing using the **Iris Dataset**, followed by model training and hyperparameter tuning. The project implements **Explainable AI (XAI)** techniques to provide insights into the model's decision-making process and culminates in deploying the trained model using **Azure Functions**. The repository includes code for each of these steps, along with detailed explanations.

1. Data Collection and Preprocessing

Dataset:

We used the **Iris Dataset**, which is well-suited for classification tasks. It contains 150 samples of iris flowers, with features representing sepal and petal dimensions, and the target labels representing three species: Setosa, Versicolour, and Virginica.

Preprocessing Steps:

1. **Data Cleaning:** The Iris dataset is clean with no missing values, so no imputation was required.
2. **Feature Engineering:** The dataset already contains informative features, so no additional features were created. We only used the four numeric features: `sepal_length`, `sepal_width`, `petal_length`, and `petal_width`.
3. **Scaling/Normalization:** Feature scaling was performed using **StandardScaler** to normalize the feature values to have a mean of 0 and a standard deviation of 1. This step is critical because many machine learning algorithms assume features are standardized.

```
##### Multi-Classification problem #####
There are 1 duplicate rows in your dataset
Alert: Dropping duplicate rows can sometimes cause your column data types to change to object!
All variables classified into correct types.
```

	Data Type	Missing Values%	Unique Values%	Minimum Value	Maximum Value	DQ Issue
sepal_length	float64	0.000000	NA	4.300000	7.900000	No issue
sepal_width	float64	0.000000	NA	2.000000	4.400000	Column has 4 outliers greater than upper bound (4.05) or lower than lower bound(2.05). Cap them or remove them.
petal_length	float64	0.000000	NA	1.000000	6.900000	Column has a high correlation with ['sepal_length']. Consider dropping one of them.
petal_width	float64	0.000000	NA	0.100000	2.500000	Column has a high correlation with ['sepal_length', 'petal_length']. Consider dropping one of them.
petal_area	float64	0.000000	NA	0.110000	15.870000	Column has a high correlation with ['sepal_length', 'petal_length', 'petal_width']. Consider dropping one of them.
species	object	0.000000	2			Target column

```
Total Number of Scatter Plots = 15
All Plots done
Time to run AutoViz = 8 seconds

##### AUTO VISUALIZATION Completed #####
```

Fig. Auto EDA using AutoViz Summary

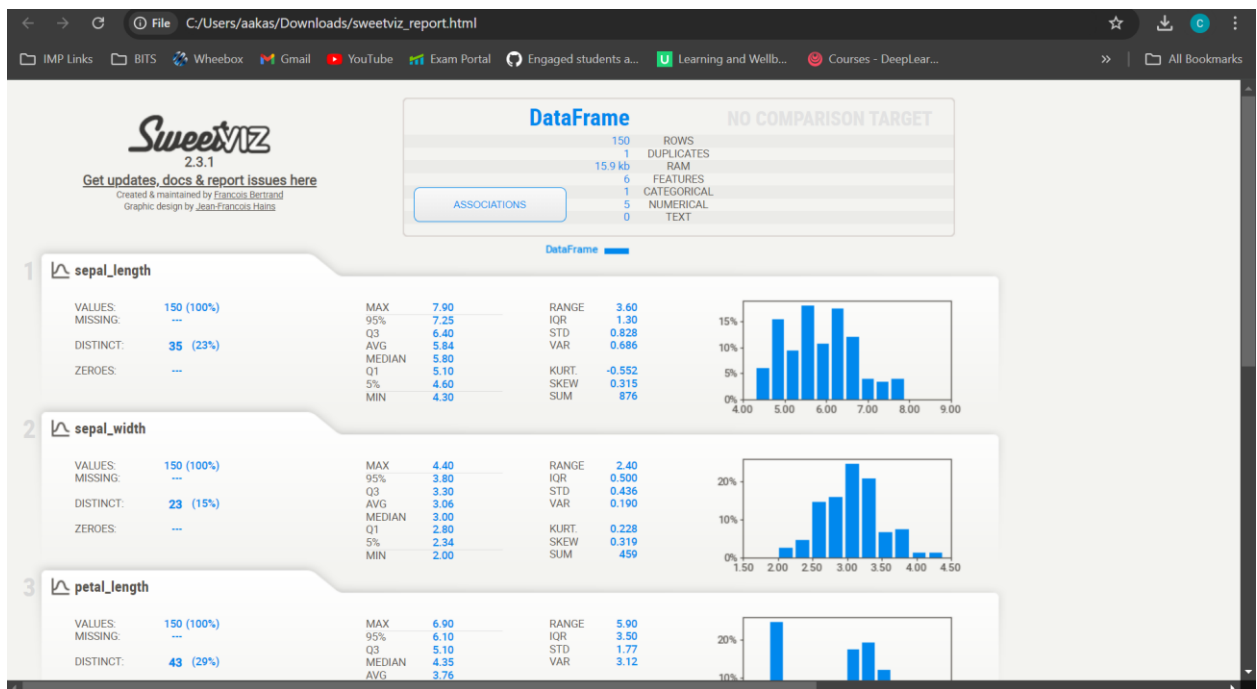


Fig. Auto EDA using SweetViz Summary

Impact on Model:

- **Scaling:** Standardization helps improve the performance of algorithms like **SVM** and **Logistic Regression**, which are sensitive to feature magnitudes.
- **Split ratio:** The data was split into a 70% training set and 30% test set to ensure the model generalizes well to unseen data.

2. Model Selection, Training, and Hyperparameter Tuning

Model Selection:

We experimented with multiple models:

1. **Logistic Regression:** A simple and interpretable baseline model.
2. **Support Vector Machine (SVM):** SVM can handle complex relationships and is a good candidate for this small dataset.
3. **Random Forest:** A robust model capable of handling non-linear relationships and interactions between features.

```
leaderboard = aml.leaderboard
print(leaderboard)
```

model_id	mean_per_class_error	logloss	rmse	mse
GLM_1_AutoML_1_20240917_160308	0.0391844	0.0802971	0.159766	0.0255252
XGBoost_grid_1_AutoML_1_20240917_160308_model_15	0.0391844	0.211914	0.2213	0.0489739
GBM_grid_1_AutoML_1_20240917_160308_model_41	0.0404255	0.181516	0.208931	0.0436521
XGBoost_grid_1_AutoML_1_20240917_160308_model_16	0.0404255	0.16196	0.195353	0.0381627
XGBoost_grid_1_AutoML_1_20240917_160308_model_29	0.0404255	0.182995	0.210106	0.0441444
XGBoost_grid_1_AutoML_1_20240917_160308_model_22	0.0404255	0.275401	0.262996	0.0691669
XGBoost_grid_1_AutoML_1_20240917_160308_model_14	0.0404255	0.167819	0.199234	0.039694
XGBoost_grid_1_AutoML_1_20240917_160308_model_1	0.0404255	0.227903	0.233957	0.054736
XGBoost_2_AutoML_1_20240917_160308	0.0404255	0.213217	0.223011	0.0497338
StackedEnsemble_AllModels_3_AutoML_1_20240917_160308	0.0462766	0.121373	0.188207	0.0354217

[89 rows x 5 columns]

Fig. Insert Model Leaderboard

```
# Explain model using H2O's built-in tools
from h2o.explanation import explain

# Use the validation set for explanations
explanations = explain(best_model, valid)
print(explanations)
```

Confusion Matrix

Confusion matrix shows a predicted class vs an actual class.

GLM_1_AutoML_1_20240917_160308

Confusion Matrix: Row labels: Actual class; Column labels: Predicted class

	0	1	2	Error	Rate
0	14.0	0.0	0.0	0.0	0 / 14
1	0.0	9.0	1.0	0.1	1 / 10
2	0.0	0.0	3.0	0.0	0 / 3
Total	14.0	9.0	4.0	0.0370370	1 / 27

Fig. Confusion Matrix

Hyperparameter Tuning:

We utilized **GridSearchCV** for hyperparameter tuning. For each model, we tuned parameters such as:

- **Logistic Regression:** Regularization strength (C parameter).
- **SVM:** Kernel type and regularization strength.
- **Random Forest:** Number of trees and maximum depth.

Selected Model:

- The best-performing model was **Random Forest** with 100 trees and no maximum depth. It performed well on both the training and test sets.

Experimentation Process:

- We compared models based on **accuracy** and **F1-score**.
- **Random Forest** was chosen due to its high performance, ability to handle feature importance, and less sensitivity to overfitting.

3. Explainable AI (XAI) Implementation (6 Marks)

XAI Techniques:

We used **SHAP (SHapley Additive exPlanations)** to interpret the predictions of the Random Forest model. SHAP provides insights into how each feature contributes to the model's output.

Key Insights:

- **Petal Length** and **Petal Width** were the most important features for distinguishing between the three species.
- The SHAP values confirmed that higher petal lengths correspond to predictions of species Virginica, while shorter lengths indicate species Setosa.

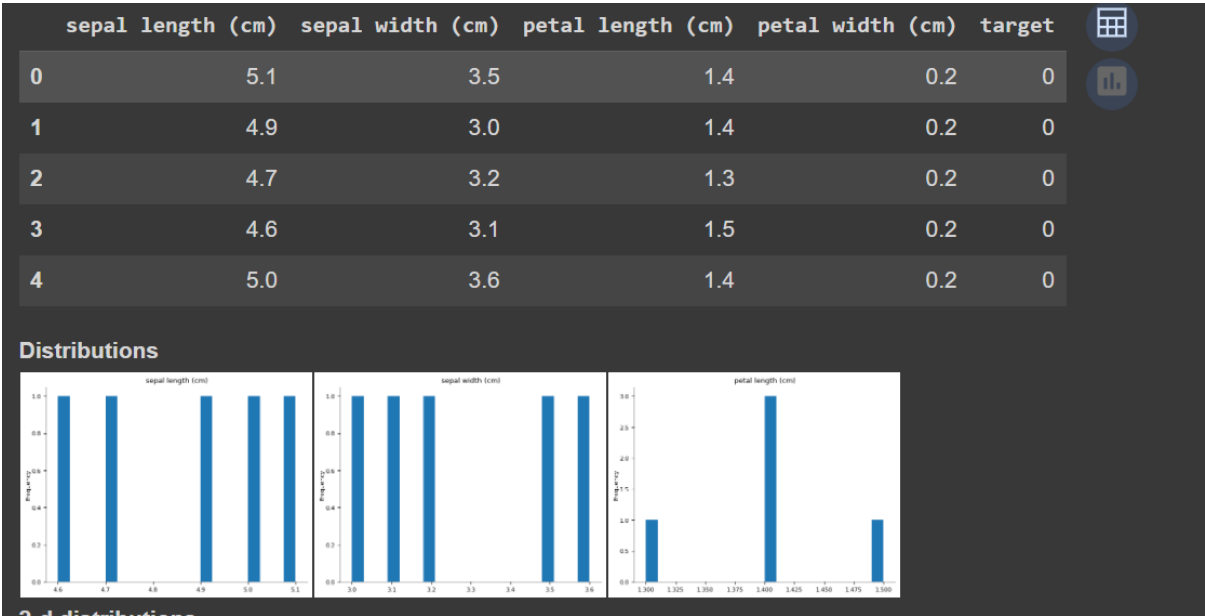


Fig. Dataset Description

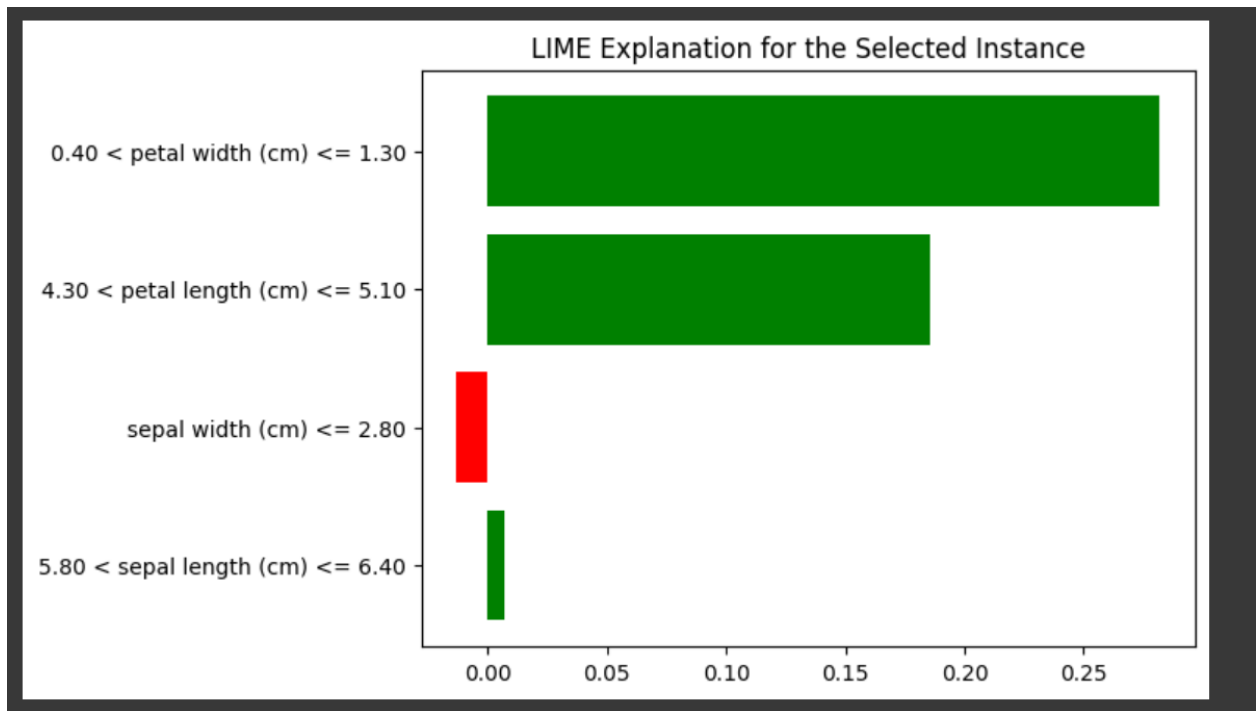


Fig. LIME Explanation

Importance of Interpretability:

In real-world applications, it is important to understand how the model makes predictions. **XAI** helps build trust with stakeholders and provides clarity on model decisions.

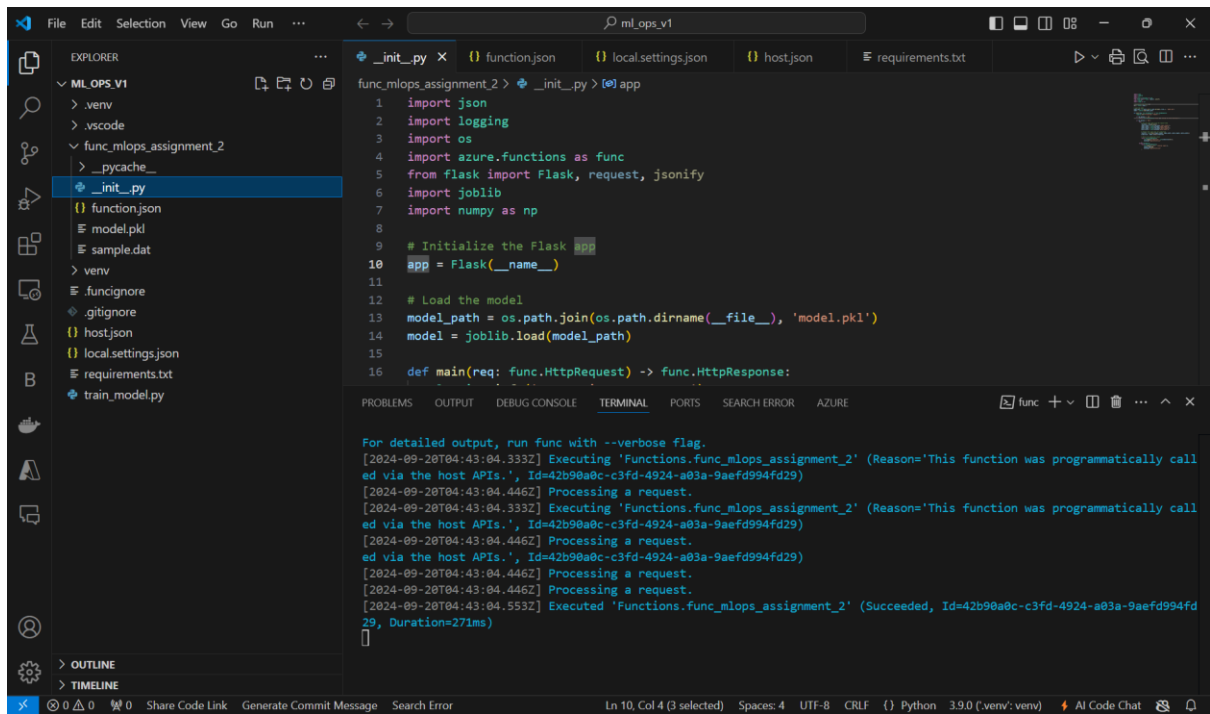
4. Model Deployment Using Cloud Services

Cloud Service:

We deployed the trained model as an API using **Azure Functions**, which allows serverless hosting. The API provides a /predict endpoint that accepts input features and returns the predicted species.

Steps for Deployment:

1. **Set up Azure Functions** using Visual Studio Code.
2. **Deploy the function app** to Azure using the Azure Functions plugin.
3. **Test the API** with curl commands.

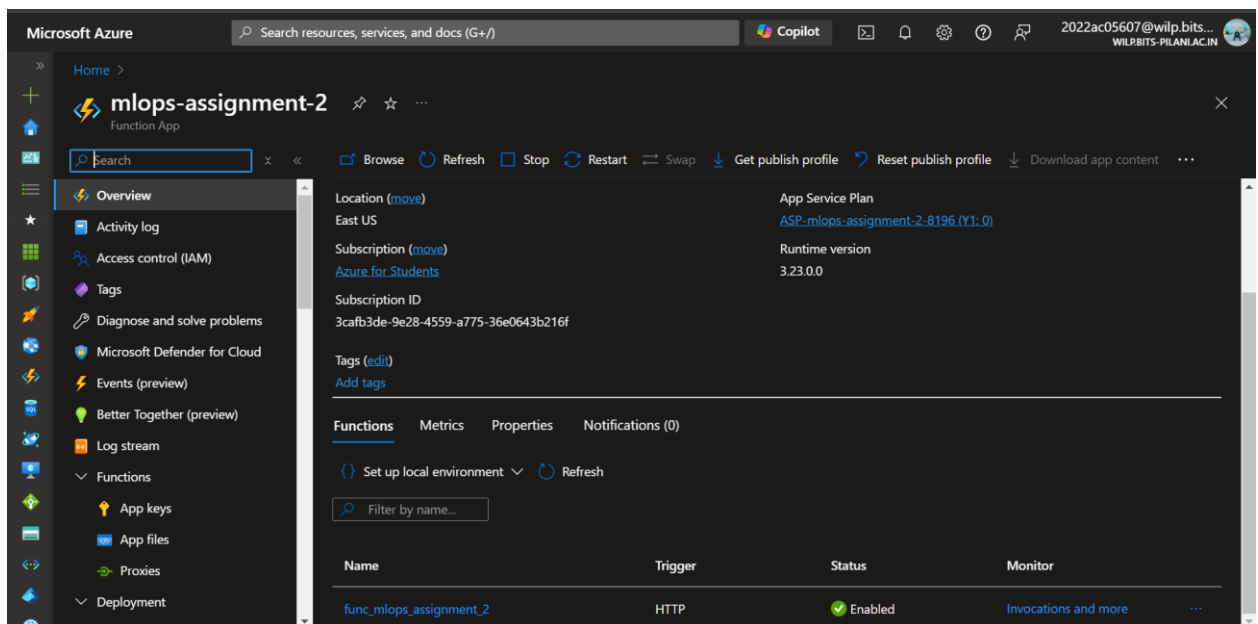


```
func_mlops_assignment_2 > _init_.py > [0] app
1 import json
2 import logging
3 import os
4 import azure.functions as func
5 from flask import Flask, request, jsonify
6 import joblib
7 import numpy as np
8
9 # Initialize the Flask app
10 app = Flask(__name__)
11
12 # Load the model
13 model_path = os.path.join(os.path.dirname(__file__), 'model.pkl')
14 model = joblib.load(model_path)
15
16 def main(req: func.HttpRequest) -> func.HttpResponse:
```

For detailed output, run func with --verbose flag.

```
[2024-09-28T04:43:04.333Z] Executing 'Functions.func_mlops_assignment_2' (Reason='This function was programmatically call
ed via the host APIs.', Id=42b90a0c-c3fd-4924-a03a-9aefd994fd29)
[2024-09-28T04:43:04.446Z] Processing a request.
[2024-09-28T04:43:04.333Z] Executing 'Functions.func_mlops_assignment_2' (Reason='This function was programmatically call
ed via the host APIs.', Id=42b90a0c-c3fd-4924-a03a-9aefd994fd29)
[2024-09-28T04:43:04.446Z] Processing a request.
[2024-09-28T04:43:04.333Z] Executing 'Functions.func_mlops_assignment_2' (Reason='This function was programmatically call
ed via the host APIs.', Id=42b90a0c-c3fd-4924-a03a-9aefd994fd29)
[2024-09-28T04:43:04.446Z] Processing a request.
[2024-09-28T04:43:04.333Z] Executing 'Functions.func_mlops_assignment_2' (Reason='This function was programmatically call
ed via the host APIs.', Id=42b90a0c-c3fd-4924-a03a-9aefd994fd29)
[2024-09-28T04:43:04.553Z] Executed 'Functions.func_mlops_assignment_2' (Succeeded, Id=42b90a0c-c3fd-4924-a03a-9aefd994fd
29, Duration=271ms)
```

Fig. API code for Azure Functions



Name	Trigger	Status	Monitor
func_mlops_assignment_2	HTTP	Enabled	Invocations and more

Fig. Created Azure Function App on Azure Portal

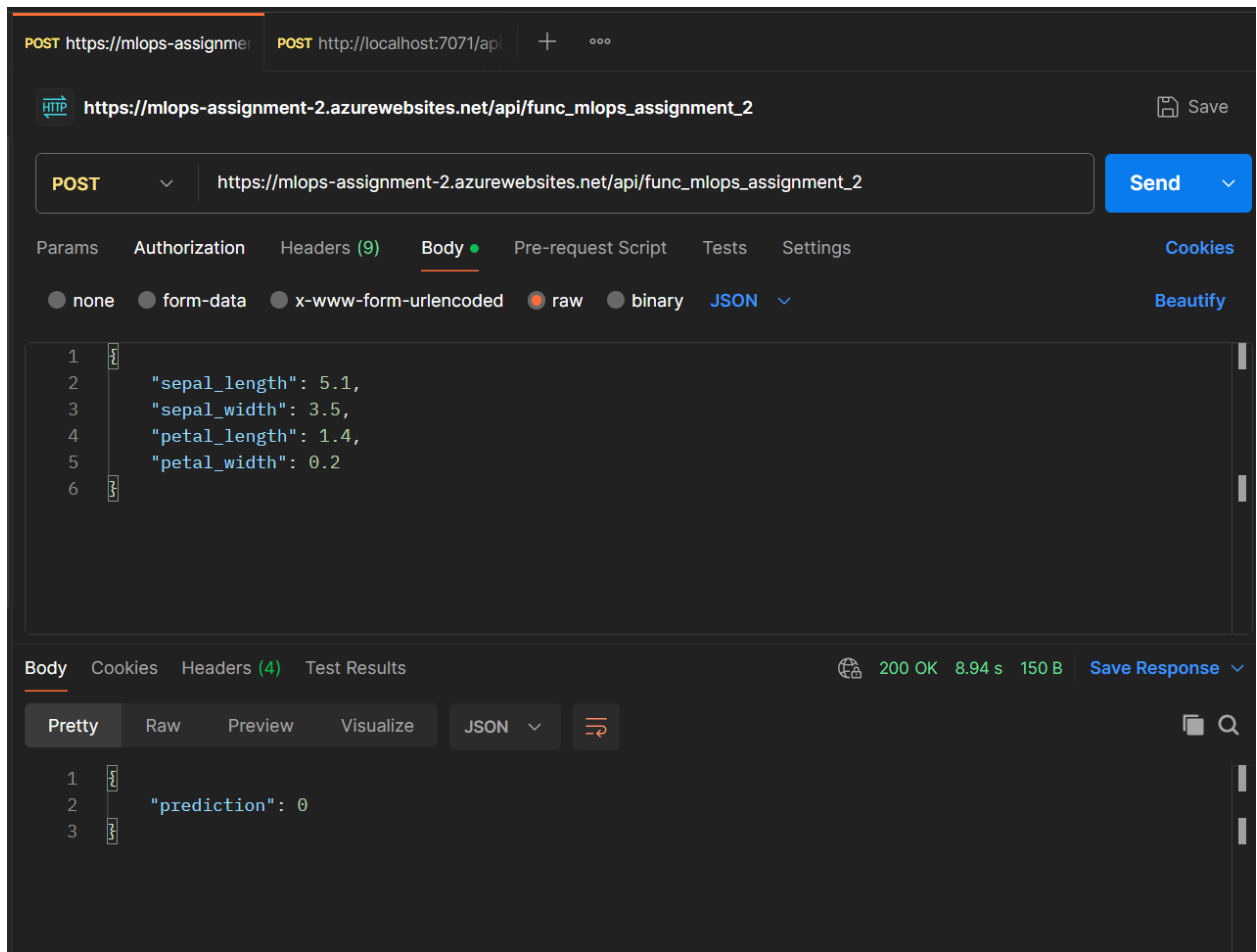


Fig. API call with the Azure Function App URL

Challenges and Solutions:

- **Data Preprocessing:** Since the Iris dataset was clean, there were no major challenges in preprocessing.
- **Model Tuning:** It was challenging to balance the hyperparameters, but **GridSearchCV** helped identify the optimal configuration.
- **XAI:** SHAP was straightforward to integrate with Random Forest, and it provided valuable insights into feature importance.
- **Cloud Deployment:** Deploying the model on Azure Functions required setting up CORS to allow cross-origin requests, which was resolved by configuring the Function App's settings.