Microservice to Expose Model via Google Colab

1. Description of the Service's General Input and Output

The service accepts a object with four numerical values representing the features of a flower from the Iris dataset. These features are:
1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

Each value should be a positive number representing the physical measurements of the flower. The input is provided in the following format:

```
{
  "input_data": [sepal_length, sepal_width, petal_length, petal_width]
}
```

The service returns a prediction in the following format, indicating the predicted flower species based on the input data. The species will be one of the following:
- Iris-setosa
- Iris-versicolor
- Iris-virginica

The output will be:

```
{
  "prediction": "Iris-setosa"
}
```

2. Specific Examples of the Service's Input and Output

Example Input:
Below is an example of the input for a flower with the following features:
- Sepal Length: 5.1 cm
- Sepal Width: 3.5 cm

- Petal Length: 1.4 cm
- Petal Width: 0.2 cm

The input will look like this:

```
{
  "input_data": [5.1, 3.5, 1.4, 0.2]
}
```

Example Output:
After the model processes this input, the service will output the following prediction:

```
{
  "prediction": "Iris-setosa"
}
```

3. Code for the Microservice in Google Colab

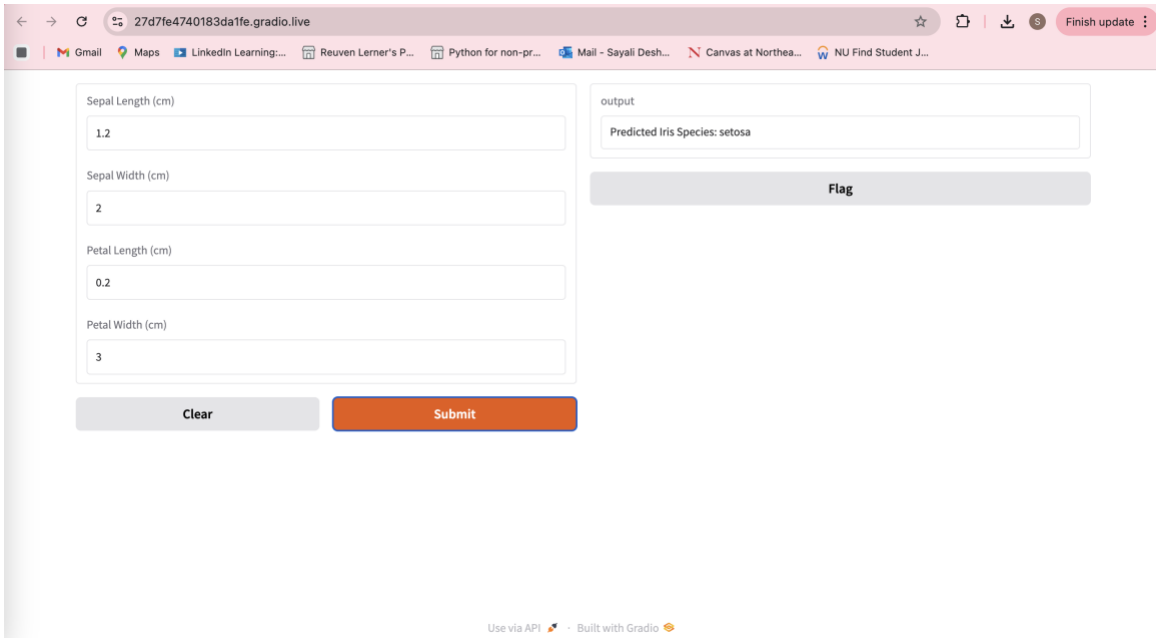
Below is the code used to create the microservice exposing the model:

```python
# Install necessary libraries
!pip install gradio

import gradio as gr
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import numpy as np

# Load dataset and train a model
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3,
random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Function to make predictions with input validation
def predict(sepal_length, sepal_width, petal_length, petal_width):
```

```
    # Validate input to ensure all values are positive numbers
    if any(val <= 0 for val in [sepal_length, sepal_width, petal_length, petal_width]):
        return "Error: All input values must be positive numbers greater than zero."

    input_data = np.array([sepal_length, sepal_width, petal_length, petal_width]).reshape(1, -1)
    prediction = model.predict(input_data)
    result = iris.target_names[prediction][0]
    return result

# Create Gradio interface
iface = gr.Interface(fn=predict, inputs=["number", "number", "number", "number"], outputs="text")

# Launch the Gradio interface and share it publicly
iface.launch(share=True)
```

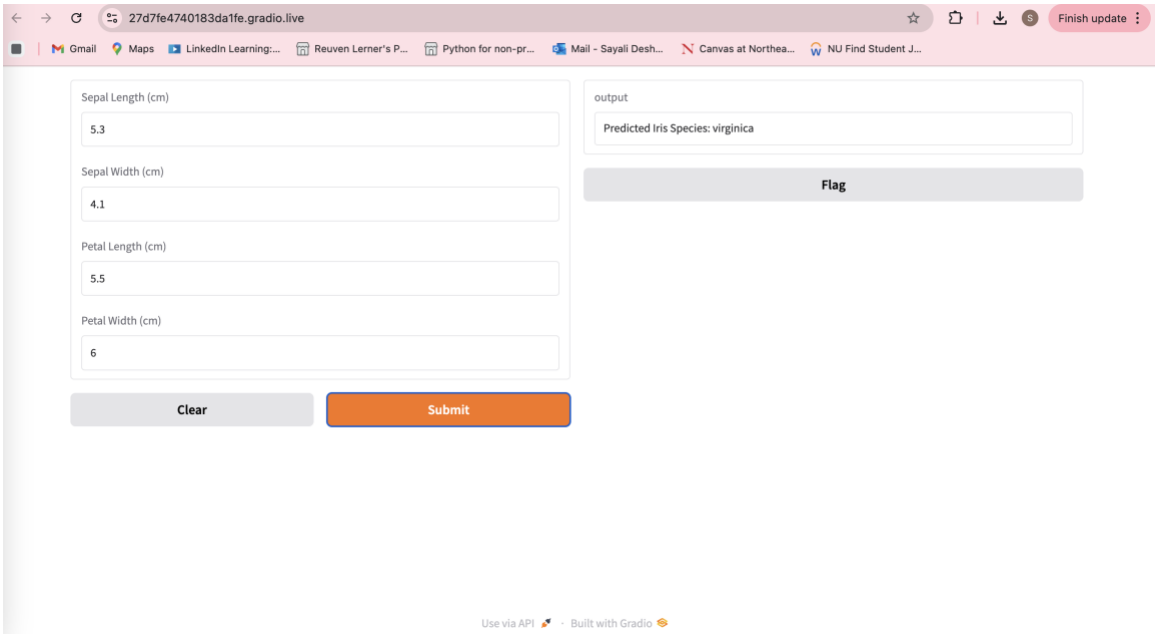
4. Screenshot(s) of a Test API Call Input and Output

Screenshot 1: Predicted Iris Species: setosa



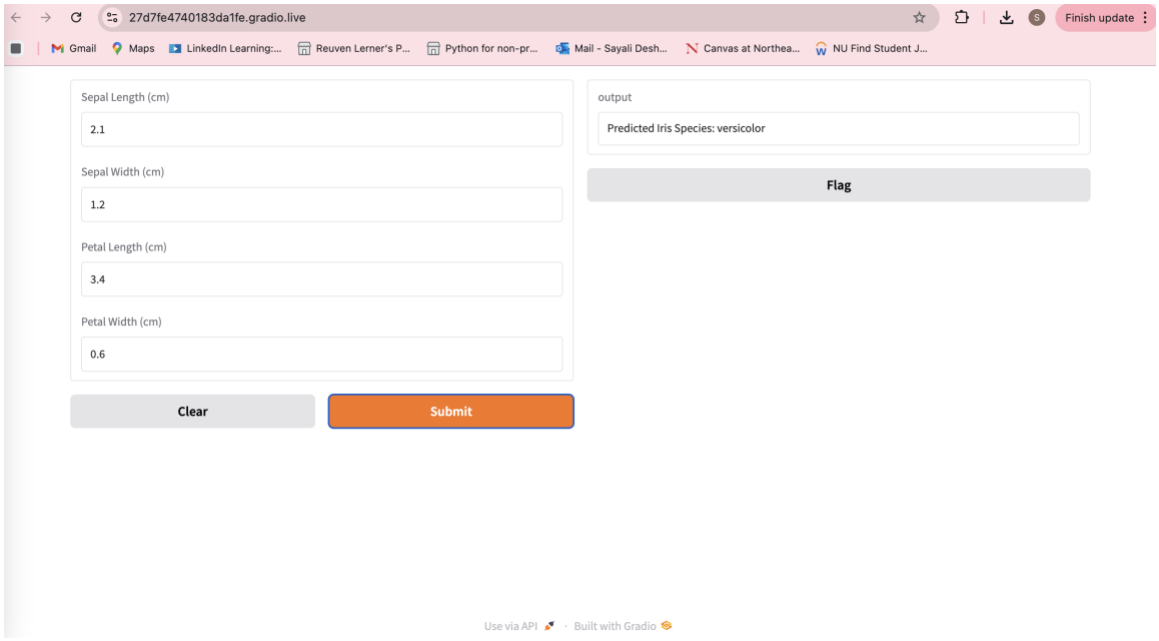
Test Input: (Sepal Length: 1.2, Sepal Width: 2, Petal Length: 0.2, Petal Width: 3)
Predicted Output: Iris Species: setosa

Screenshot 2: Predicted Iris Species: virginica



Test Input: (Sepal Length: 5.3, Sepal Width: 4.1, Petal Length: 5.5, Petal Width: 6)
Predicted Output: Iris Species: virginica.

Screenshot 3: Predicted Iris Species: versicolor



Test Input: (Sepal Length: 2.1, Sepal Width: 1.2, Petal Length: 3.4, Petal Width: 0.6)
Predicted Output: Iris Species: versicolor.

5. URL of the Service

The Gradio interface is publicly available at the following URL:

https://4256730696347065af.gradio.live

You can visit this URL to input values and get predictions for the flower species.

Conclusion

In this report, a microservice was created using Google Colab and Gradio to expose a trained machine learning model. The model predicts flower species based on input values for sepal and petal measurements. The service was tested, and input-output examples were provided. The URL of the service was shared for easy access.

Reference

Lee, Dr. E. (2023, September 25). *Creating and deploying a neural network with Google Colab, flask, github, and gitpod as a Docker...* Medium. https://drlee.io/creating-and-deploying-a-neural-network-with-google-colab-flask-github-and-gitpod-as-a-docker-fc4a5f384573

How to manually test a web API. (n.d.-a). https://how-to.dev/how-to-manually-test-a-web-api