PDF Layout – Week 5: Cloud and API Deployment

Title: Week 5: Cloud and API Deployment

Name: Radhika Ranchhodbhai Diyora
Batch Code: LISUM51
Submission Date: 4 December 2025
Submitted To: Data Glacier

---

1. Project Overview

This project demonstrates deployment of a trained Iris classification model as a **web application** and **API** using Flask and Render (cloud).

Features:

- Users can predict Iris species using a web form.
- API endpoint allows JSON input for programmatic predictions.
- Deployment is done using **Docker** on **Render**.

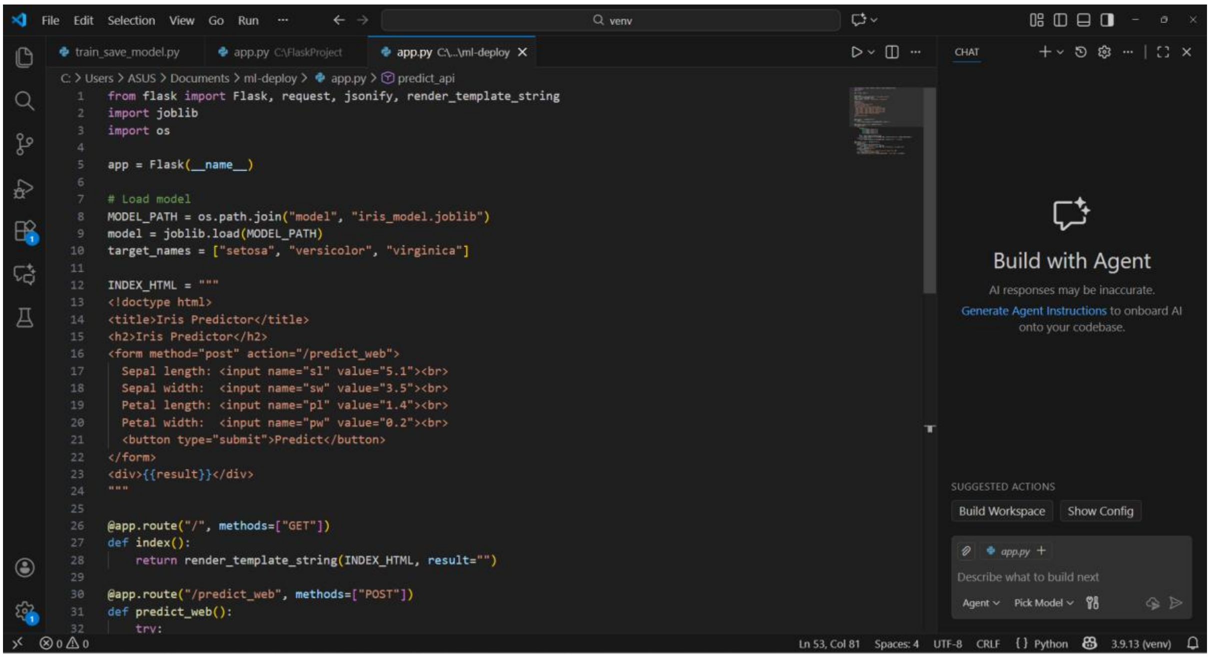
---

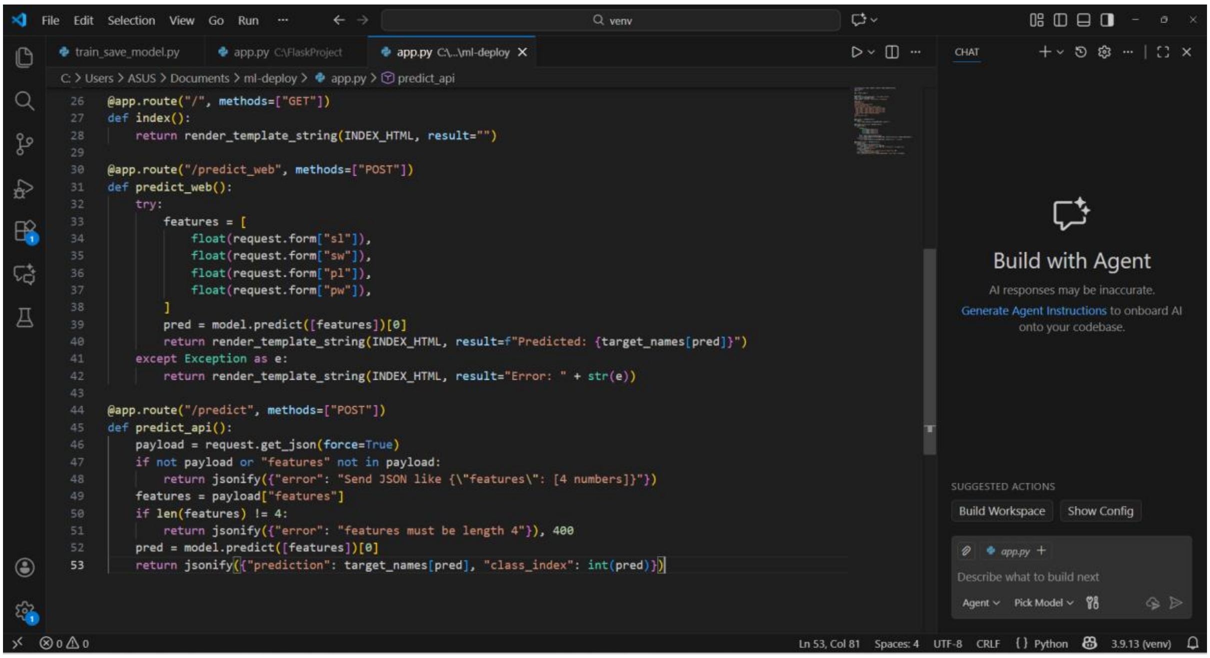
2. Steps of Deployment

Step 1 – Prepare Model & Flask App

- Trained Iris model saved as model/iris_model.joblib.
- Flask app (app.py) created with:
  - Web form endpoint / and /predict_web
  - API endpoint /predict

Screenshot 1: *(screenshot of app.py showing Flask routes here)*





Step 2 – Dockerize the App

- Created a Dockerfile with required packages and **start command** using gunicorn.

Screenshot 2: *(screenshot of Dockerfile content showing CMD with $PORT here)*

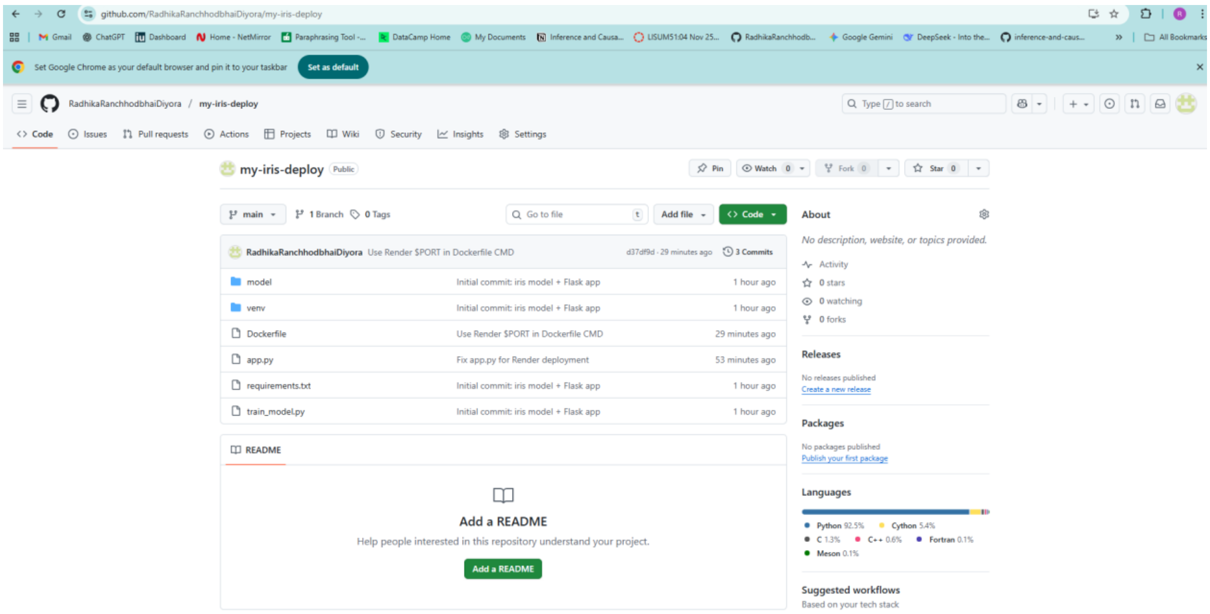
```
# Use official Python slim image
FROM python:3.11-slim

# Set working directory
WORKDIR /app

# Copy requirements and install
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy all files
COPY . .

# Expose port (optional, Render uses $PORT)
EXPOSE 8080

# Start the Flask app using Render's assigned $PORT
CMD ["sh", "-c", "gunicorn -w 4 -b 0.0.0.0:$PORT app:app"]
```

Step 3 – Push Code to GitHub

- All code pushed to:

https://github.com/RadhikaRanchhodbhaiDiyora/my-iris-deploy

Screenshot 3: *(screenshot of GitHub repo showing files and latest commit here)*

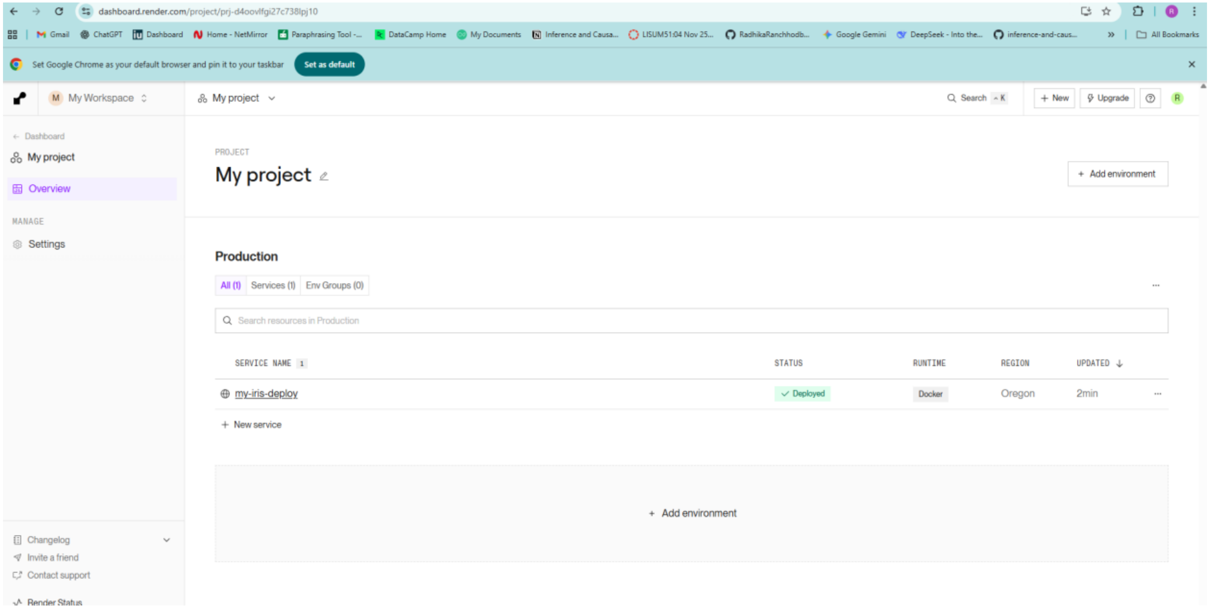


Step 4 – Deploy on Render

- Connected GitHub repo to Render.
- Render built Docker image and started web service.
- URL of deployed app:

https://my-iris-deploy.onrender.com

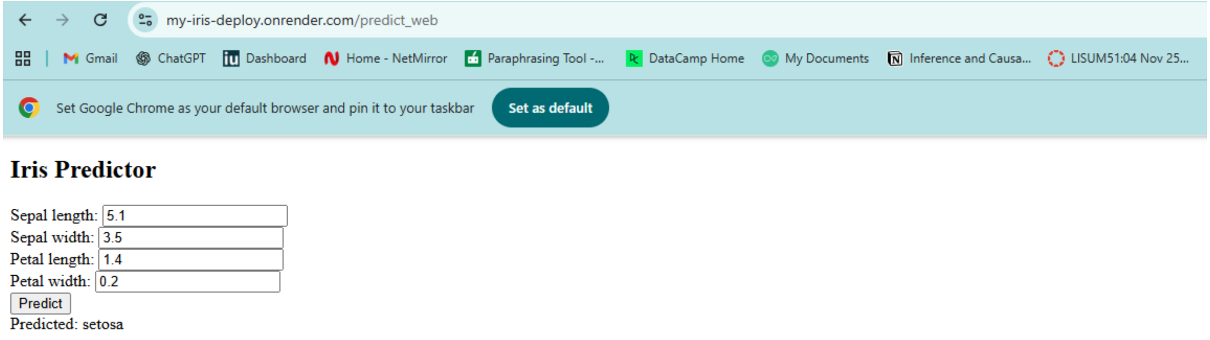
Screenshot 4: *(screenshot of Render dashboard showing deployed service and logs here)*



Step 5 – Test Web App

- Open URL in browser.
- Enter sample features:
  - Sepal length: 5.1
  - Sepal width: 3.5
  - Petal length: 1.4
  - Petal width: 0.2
- Click **Predict** → result displayed.

Screenshot 5: *(screenshot of browser showing HTML form and prediction result here)*



Step 6 – Test API

- Command used in terminal (CMD/PowerShell):

curl -X POST -H "Content-Type: application/json" -d "{\"features\": [5.1,3.5,1.4,0.2]}" https://my-iris-deploy.onrender.com/predict

- Response:

{"class_index":0,"prediction":"setosa"}

Screenshot 6: *(screenshot of CMD showing API JSON response here)*

---

3. Conclusion

- Successfully deployed a machine learning model as a **web app and API**.
- Both **web interface** and **API endpoint** are working as expected.
- All files and deployment can be verified via **GitHub repo** and **Render URL**.