

Week 5: Cloud and API Deployment

Name: Deployment on Flask

Report date: 05-June-2025

Internship Batch: LISUM45 Version:1.0

Data intake by: Najma Abdi Mohamed

Data intake reviewer : Data Glacier

Data storage location:

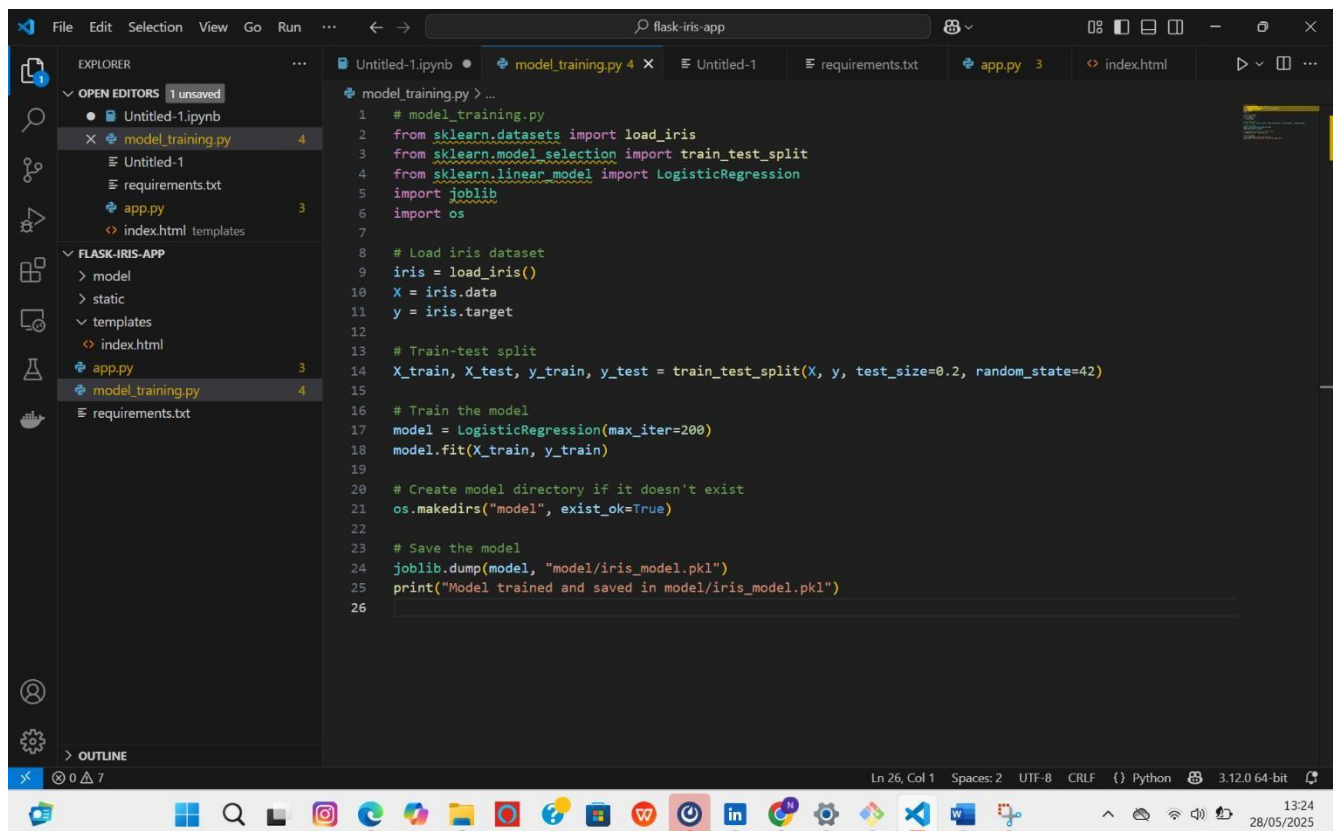
<https://github.com/najmaabdi99/Data-Glacier-Internship---Flask-App-Deployment.git>

Tabular data details:

Total number of observations	150 rows (each for one flower sample)
Total number of files	1
Total number of features	4
Base format of the file	.csv
Size of the data	5 KB

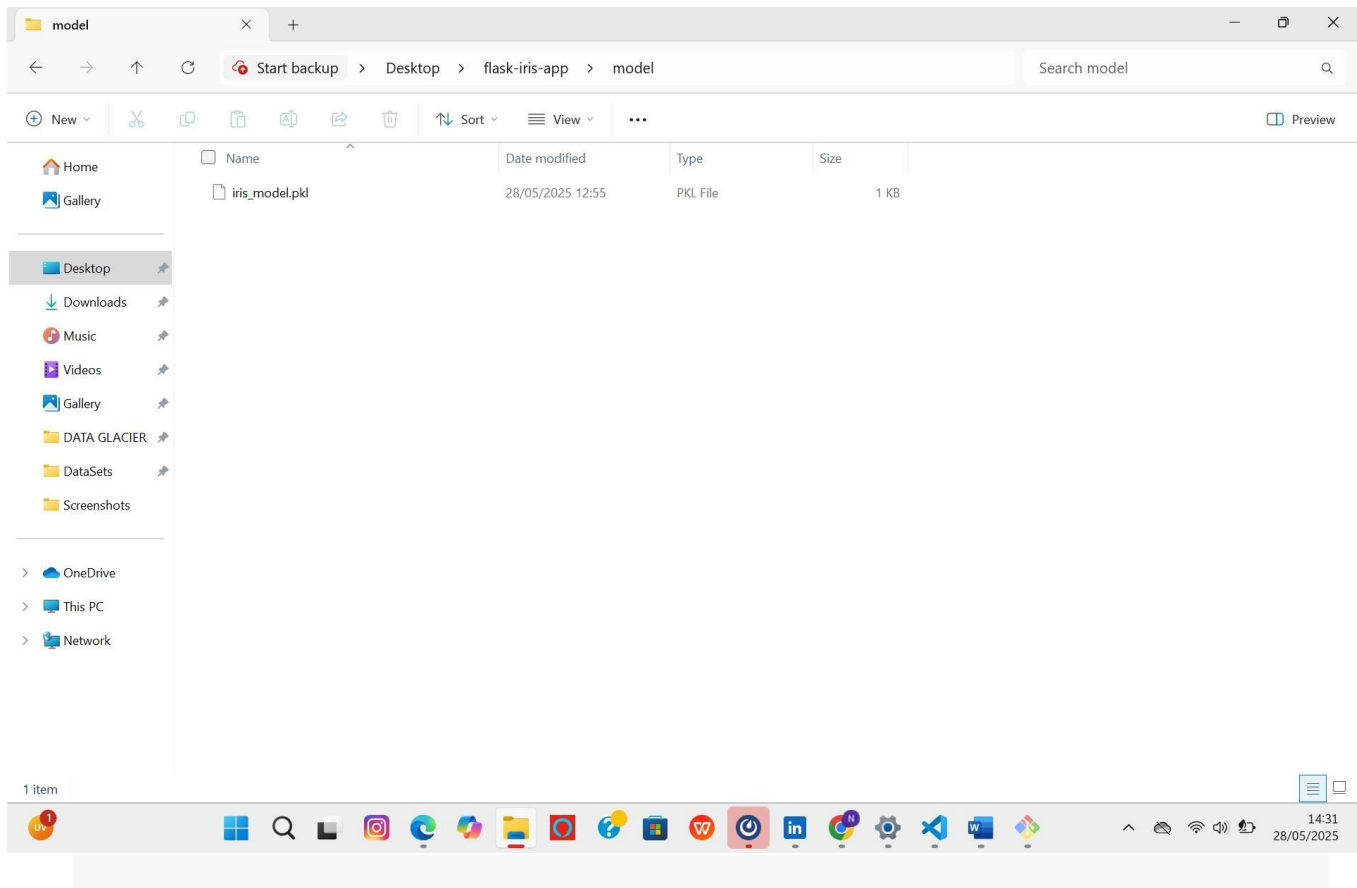
1- Building Model Training and Saving.

Here, iris dataset was loaded, train-test split done, model trained and model saved.



```
1 # model_training.py
2 from sklearn.datasets import load_iris
3 from sklearn.model_selection import train_test_split
4 from sklearn.linear_model import LogisticRegression
5 import joblib
6 import os
7
8 # Load iris dataset
9 iris = load_iris()
10 X = iris.data
11 y = iris.target
12
13 # Train-test split
14 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
15
16 # Train the model
17 model = LogisticRegression(max_iter=200)
18 model.fit(X_train, y_train)
19
20 # Create model directory if it doesn't exist
21 os.makedirs("model", exist_ok=True)
22
23 # Save the model
24 joblib.dump(model, "model/iris_model.pkl")
25 print("Model trained and saved in model/iris_model.pkl")
26
```

1.2 Saved Model



2- Deploying The Model on Flask (Web App)

2.1 app.py

This is the main Flask backend file. It:

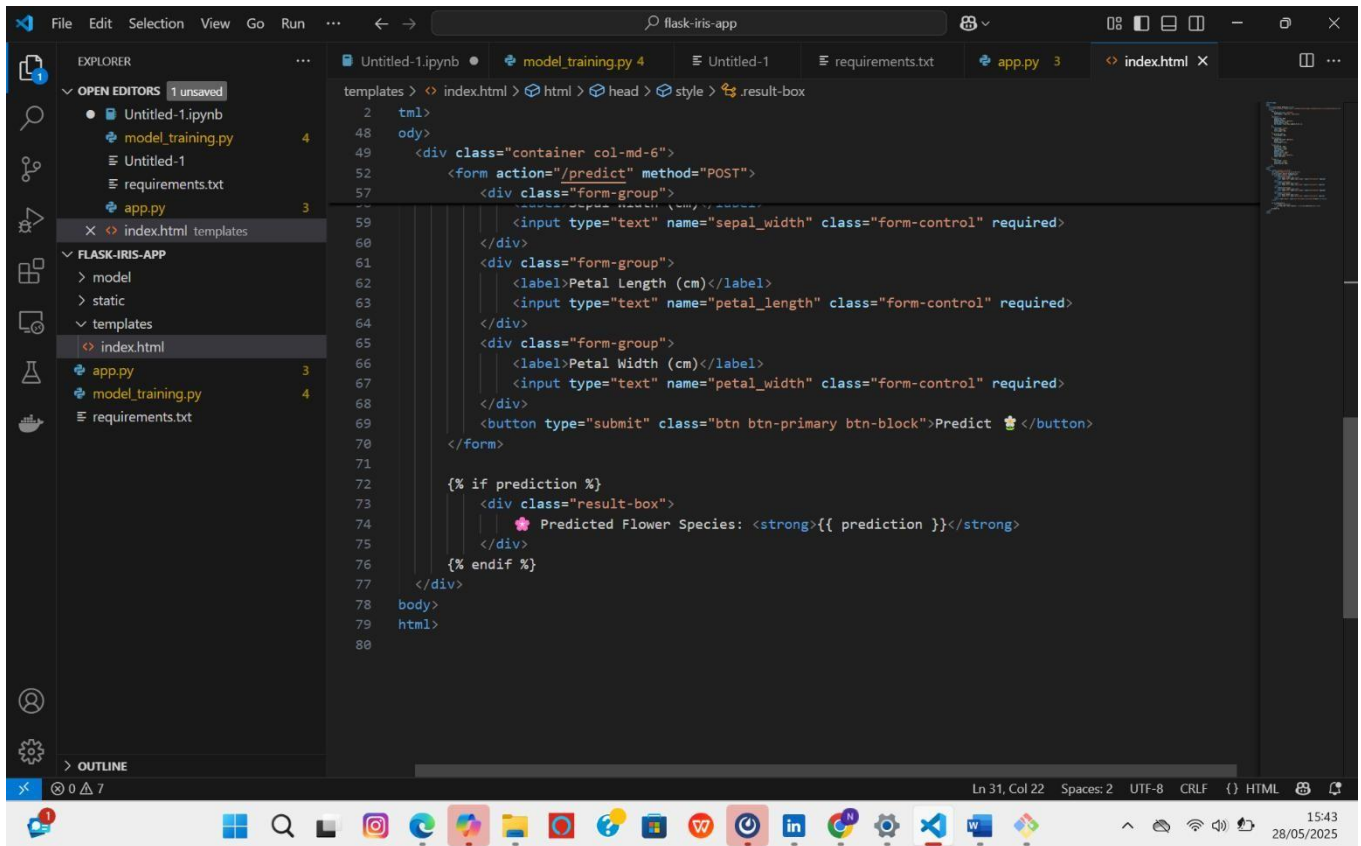
- Loads the saved model (model/iris_model.pkl)
- Sets up routes (/ for form, /predict for predictions)
- Handles user input and displays the prediction result

```
1 from flask import flask, request, render_template
2 import joblib
3 import numpy as np
4
5 app = Flask(__name__)
6 model = joblib.load("model/iris_model.pkl")
7
8 @app.route('/')
9 def home():
10     return render_template("index.html")
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14     if request.method == 'POST':
15         sl = float(request.form['sepal_length'])
16         sw = float(request.form['sepal_width'])
17         pl = float(request.form['petal_length'])
18         pw = float(request.form['petal_width'])
19
20         data = np.array([[sl, sw, pl, pw]])
21         class_index = model.predict(data)[0]
22
23         # Map numeric class to flower name
24         flower_names = ['Setosa 🌸', 'Versicolor 🌸', 'Virginica 🌸']
25         prediction = flower_names[class_index]
26
27         return render_template("index.html", prediction=prediction)
28
29 if __name__ == '__main__':
30     app.run(debug=True)
```

2.2 index.html

This is the frontend HTML page that:

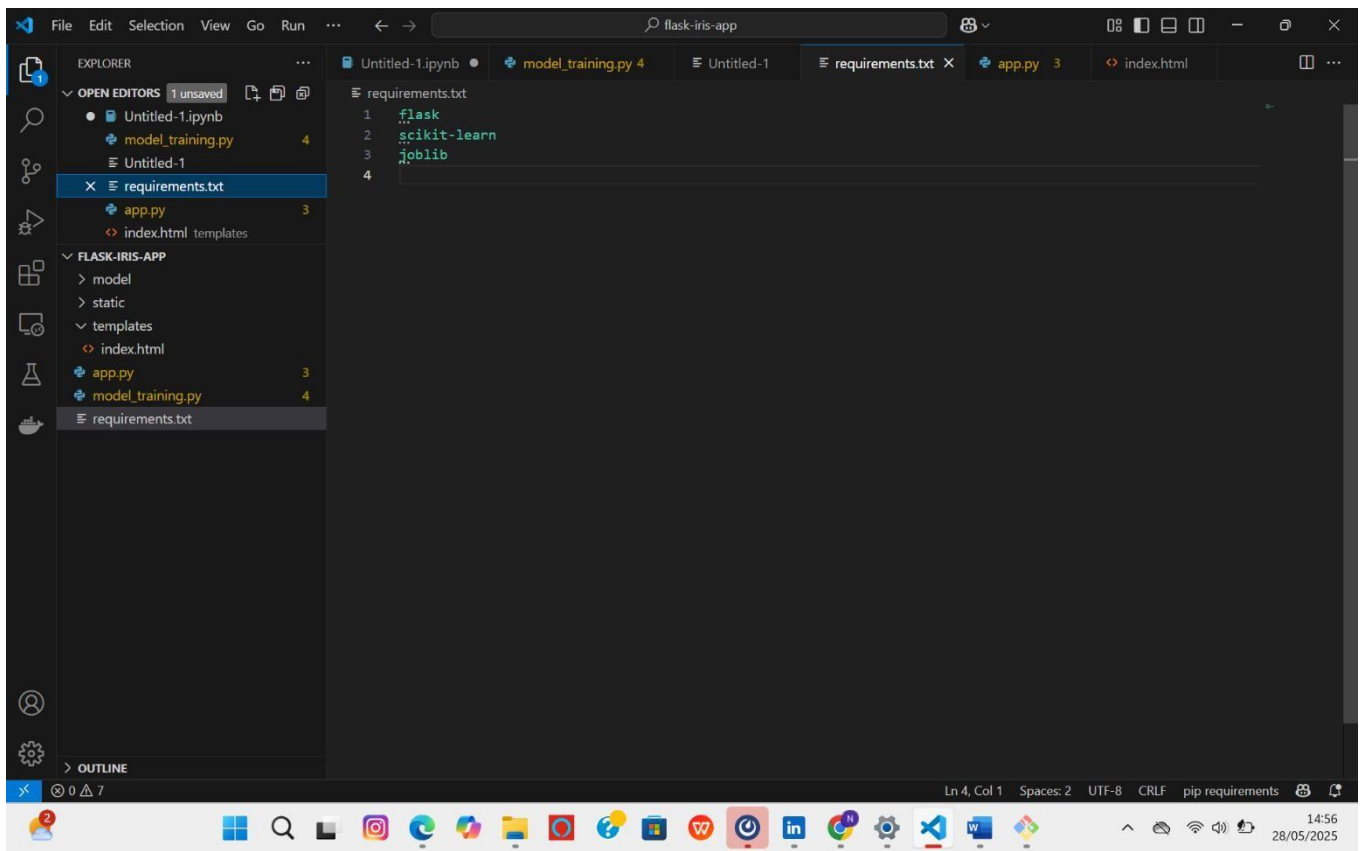
- Collects user input for sepal/petal measurements
- Sends it to Flask via a POST request
- Displays the predicted iris class on the same page



2.3 requirements.txt

This file lists all the Python packages needed to run the app, such as:

- Flask for the web server
- scikit-learn for the model
- joblib for loading the saved model




2.4 Running Flask Code

```
MINGW64/c:/Users/abdin/OneDrive/Desktop/flask-iris-app
usage: conda-script.py [-h] [-no-plugins] [-v] COMMAND ...
conda-script.py: error: argument COMMAND: invalid choice: 'activate' (choose from 'clean', 'compare', 'config', 'create', 'info', 'init', 'install', 'list', 'no
tices', 'package', 'remove', 'uninstall', 'rename', 'run', 'search', 'update', '
upgrade', 'build', 'content-trust', 'convert', 'debug', 'develop', 'doctor', 'in
dex', 'inspect', 'metapackage', 'render', 'skeleton', 'repo', 'env', 'pack', 'to
ken', 'verify', 'server')
(base)
abdin@NAJMA MINGW64 ~/OneDrive/Desktop/flask-iris-app (master)
$ python app.py
Traceback (most recent call last):
  File "C:\Users\abdin\OneDrive\Desktop\flask-iris-app\app.py", line 6, in <modu
le>
    model = joblib.load("iris_model.pkl") # Ensure this matches your model file
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 720-665-916
127.0.0.1 - - [28/May/2025 15:38:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2025 15:38:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2025 15:38:31] "POST /predict HTTP/1.1" 200 -
* Detected change in 'C:\Users\abdin\OneDrive\Desktop\flask-iris-app\app.py', reloading
* Detected change in 'C:\Users\abdin\OneDrive\Desktop\flask-iris-app\app.py', reloading
* Restarting with watchdog (windowsapi)
(base)
abdin@NAJMA MINGW64 ~/OneDrive/Desktop/flask-iris-app (master)
$ python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 720-665-916
127.0.0.1 - - [28/May/2025 15:41:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2025 15:42:04] "POST /predict HTTP/1.1" 200 -
```

Home Page - Select or create a notebook | Untitled - Jupyter Notebook | Iris Flower Predictor

127.0.0.1:5000



Iris Flower Species Predictor

Sepal Length (cm)

Sepal Width (cm)

Petal Length (cm)

Petal Width (cm)

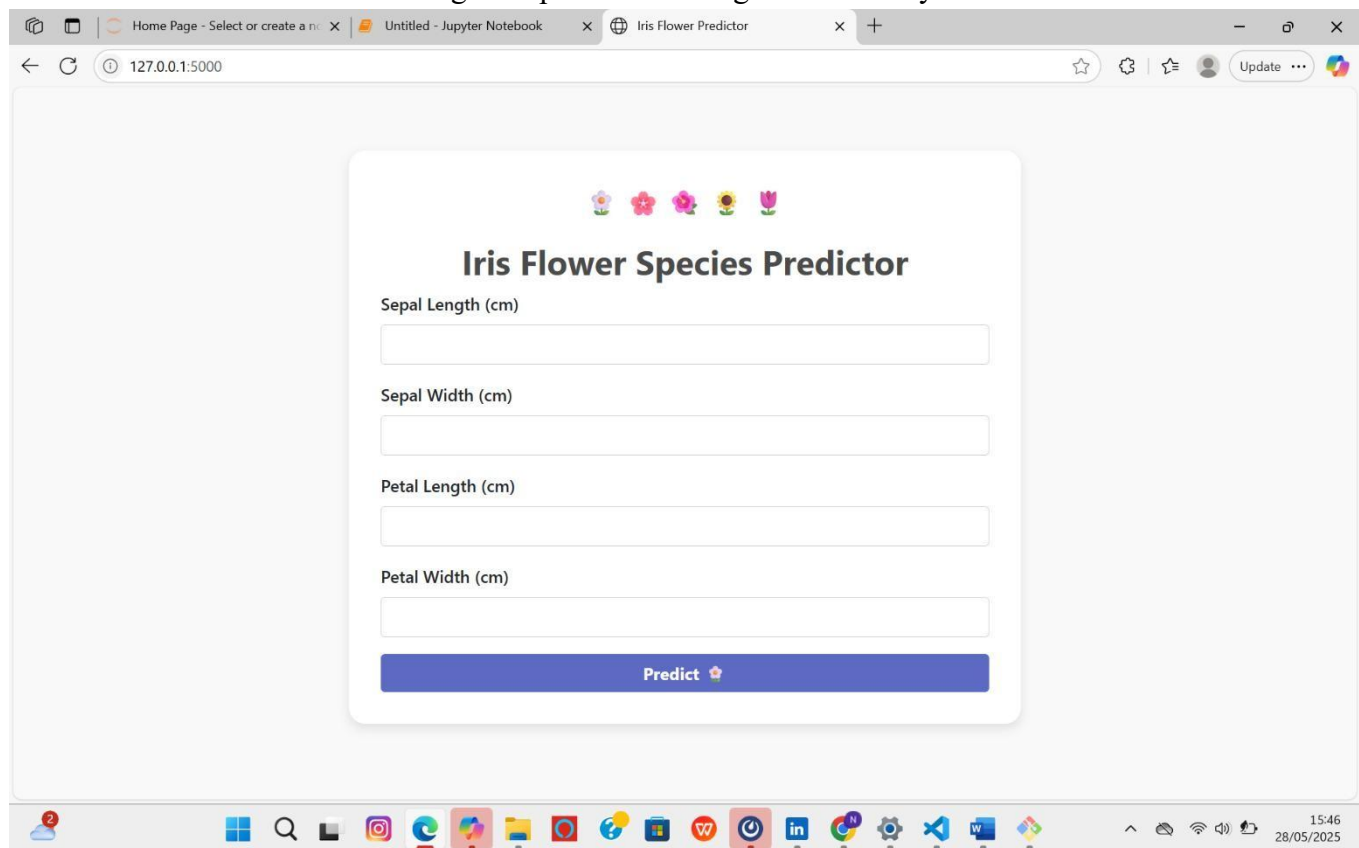
Predict 🌸

2.5 Prediction Output

This section showcases the final output of the web application after submitting user input through the form.

1. After entering flower feature values (sepal and petal measurements) into the form on the homepage (index.html), the user clicks the "Predict" button.
2. The input is sent to the Flask backend, where the model makes a prediction.
3. The predicted Iris flower species (e.g., Setosa, Versicolor, or Virginica) is displayed dynamically on the same page below the form.

This confirms the model is working as expected and integrated correctly with the web interface.



The screenshot displays a web browser window with the address bar showing '127.0.0.1:5000'. The browser tabs include 'Home Page - Select or create a n...', 'Untitled - Jupyter Notebook', and 'Iris Flower Predictor'. The main content area features a white card with a light gray border. At the top of the card, there are five flower emojis: a purple one, a pink one, a yellow one, a red one, and a blue one. Below the emojis, the title 'Iris Flower Species Predictor' is centered in a bold, black font. Underneath the title, there are four input fields, each with a label to its left: 'Sepal Length (cm)', 'Sepal Width (cm)', 'Petal Length (cm)', and 'Petal Width (cm)'. Each input field is a simple white rectangle with a thin gray border. At the bottom of the card, there is a prominent blue button with the text 'Predict' in white, followed by a small flower emoji.

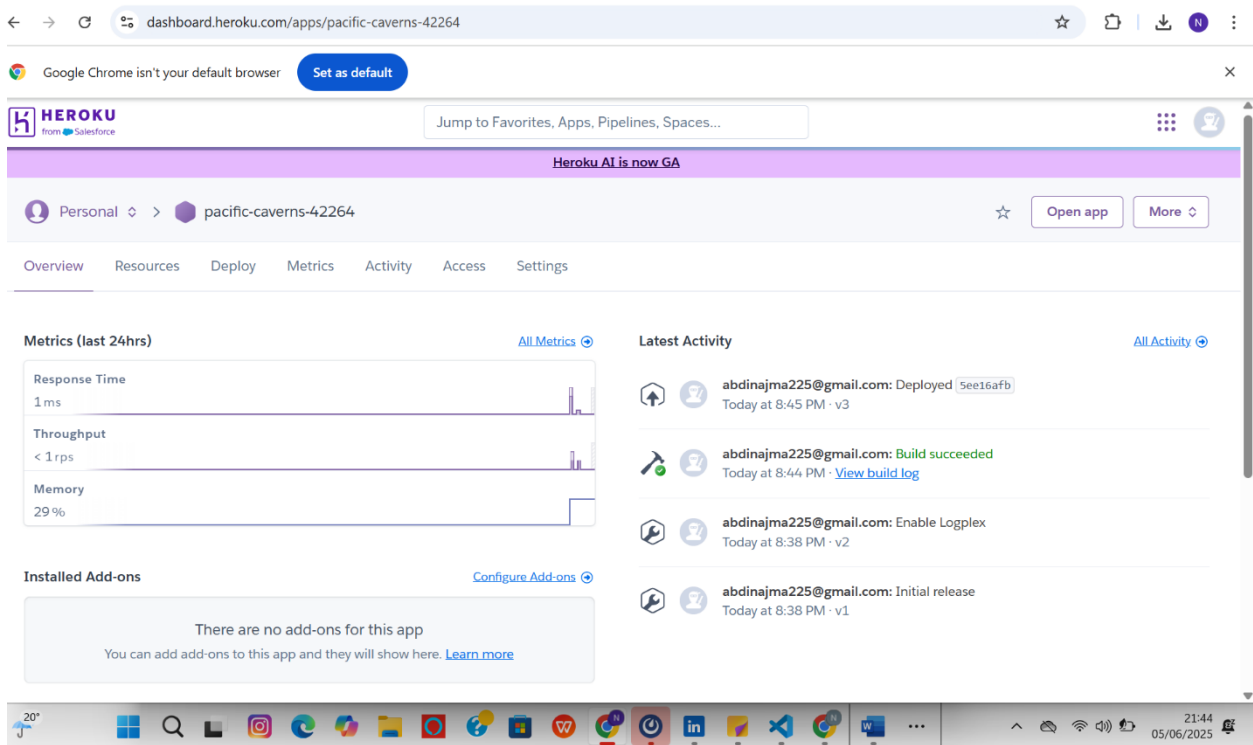
5. Model deployment using Heroku

We're ready to start our Heroku deployment now that our model has been trained, the machine learning pipeline has been set up, and the application has been tested locally. To upload the application source code onto Heroku i went through the following process

- ☐ Initialize a Git repo in your project folder
- ☐ Add Heroku as a remote.
- ☐ Commit code locally.
- ☐ Push my code to Heroku's remote repo.

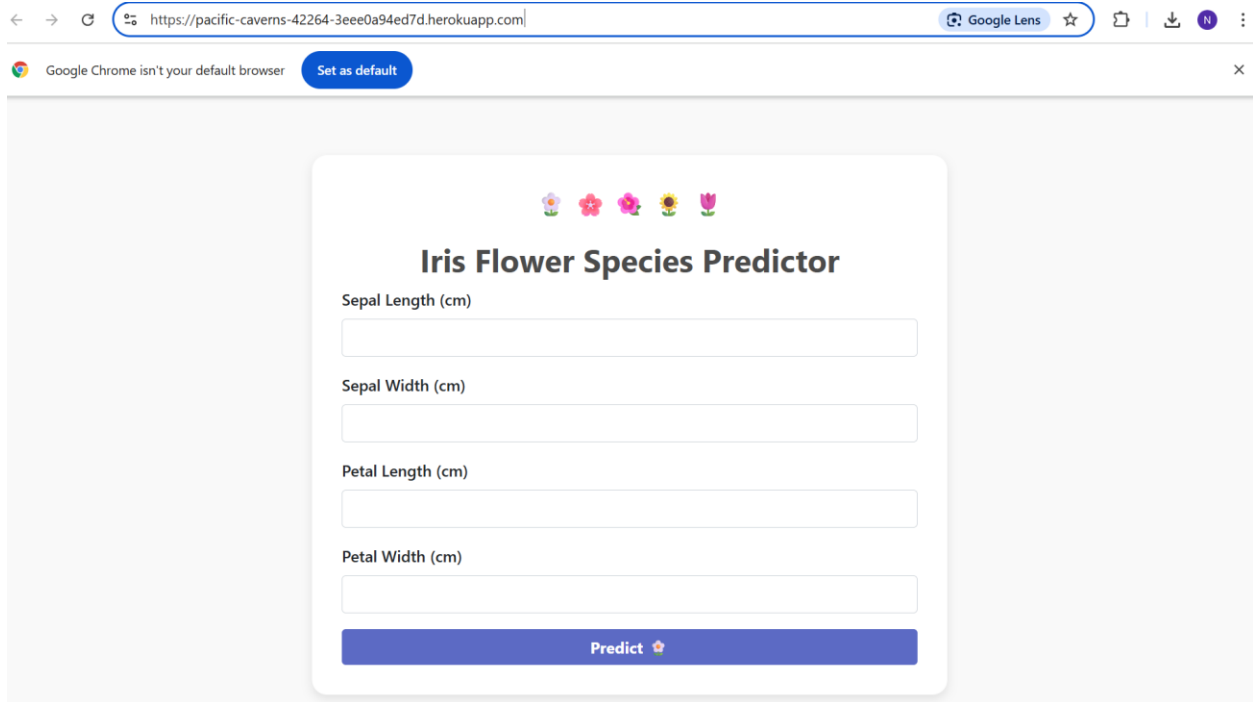
3.1 Cloud Dashboard Screenshot

This screenshot shows the Heroku Dashboard with the app name (pacific-caverns-42264) and status (Deployed). It confirms that the app is deployed on Heroku.



3.2 APP URL SCREENSHOT

This screenshot shows the app's landing page opened in a browser using the URL provided by Heroku. It confirms that the app is live and accessible.



The screenshot shows a web browser window with the address bar displaying the URL `https://pacific-caverns-42264-3eee0a94ed7d.herokuapp.com/`. The browser interface includes navigation buttons, a search bar with "Google Lens", and a notification that "Google Chrome isn't your default browser" with a "Set as default" button. The main content area features a white card titled "Iris Flower Species Predictor" with five flower icons at the top. The card contains four input fields for "Sepal Length (cm)", "Sepal Width (cm)", "Petal Length (cm)", and "Petal Width (cm)". A blue "Predict" button with a flower icon is at the bottom of the card.

Iris Flower Species Predictor

Sepal Length (cm)

Sepal Width (cm)

Petal Length (cm)

Petal Width (cm)

Predict 🌸